TAMPERE UNIVERSITY OF TECHNOLOGY

PAULI RAUTAKORPI

MATHEMATICAL MODELING OF KITE GENERATORS

Master of Science Thesis

Examiners: Professor Keijo Ruohonen
Lecturer Risto Silvennoinen
Examiner and subject approved in
the Faculty of Natural Sciences
Council meeting on 8 May 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Science and Engineering
RAUTAKORPI, PAULI: Mathematical modeling of kite generators
Master of Science Thesis, 59 pages, 20 Appendix pages
December 2013
Major: Mathematics
Examiners: Professor Keijo Ruohonen, Lecturer Risto Silvennoinen
Keywords: kite generator, kite, electricity generation, wind energy

A kite wind generator is a new type of wind power plant that uses a kite or a tethered wing to extract energy from wind. In this thesis a so-called pumping kite generator is considered that has a working cycle with alternating energy generation and retraction phases. In the energy generation phase the kite is flown so that it pulls the tether out of the ground station with maximal power. The tether is wound on a drum attached to the generator in the ground station and thus the generator rotates and generates electricity when the kite pulls the tether. In the retraction phase the pulling force of the kite is minimized and the tether is reeled back in around the drum with minimal loss of generated energy.

This thesis presents the theory needed for simulating such pumping kite generator during the energy generation phase. The model presented is a simple point mass model with some more advanced features including two options for control angle definition and the effect of elasticity and tether drag on the real distance between the ends of the tether. Also a working example simulator implementation on MATLAB is presented with the source code attached. All the equations are derived with detailed analysis and also all the details of the simulator implementation are documented.

In the end this kite simulator is used to analyze kite trajectories with constant tether length and pre-defined sinusoidal periodic control function. The results are very interesting with some stable orbits found so that the kite can stabilize on them without any feedback and continue flying on them until the conditions change. These stable orbits have an useful lying-eight shape and might be useful in kite generators to control the kite easier with the help of natural stability.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Teknis-luonnontieteellinen koulutusohjelma
RAUTAKORPI, PAULI: Leijavoimalan matemaattinen mallintaminen
Diplomityö, 59 sivua, 20 liitesivua
Joulukuu 2013
Pääaine: Matematiikka
Tarkastajat: professori Keijo Ruohonen, lehtori Risto Silvennoinen
Avainsanat: leijavoimala, leija, sähköntuotanto, tuulienergia

Leijavoimala on uudentyyppinen tuulivoimala, jossa energiaa kerätään tuulesta leijaa lennättämällä. Tässä työssä esitellään pumppaava leijavoimalatyyppi, jonka toiminta perustuu vuorotteleviin sähköntuotantovaiheeseen ja paluuvaiheeseen. Energiantuotantovaiheessa leija vetää maassa olevaa generaattoria pyörittävän rummun ympärille kelattua köyttä mahdollisimman suurella voimalla pois rummulta, jolloin generaattori tuottaa pyöriessään sähköä. Paluuvaiheessa puolestaan leijan vetovoima minimoidaan ja köysi kelataan takaisin rummun ympärille käyttämällä generaattoria moottorina.

Tämä työ esittelee tällaisen pumppaavan leijavoimalan energiantuotantovaiheen simuloimiseen tarvittavan teorian. Esiteltävä malli on yksinkertainen massapistemalli, jossa on joitakin edistyneempiä ominaisuuksia kuten kaksi vaihtoehtoista ohjauskulman määritelmää sekä köyden venymän ja ilmanvastuksen vaikutus köyden päiden todelliseen välimatkaan. Työ esittelee myös MATLABilla toteutetun toimivan simulaattorin, jonka lähdekoodi on liitteenä. Kaikki yhtälöt johdetaan yksityiskohtaisesti ja myös kaikki simulaattorin toteutuksen yksityiskohdat dokumentoidaan.

Lopussa tätä leijasimulaattoria käytetään leijan lentoratojen tutkimiseen vakiopituisella köydellä ja ennalta määritellyllä sinimuotoisella jaksollisella ohjausfunktiolla. Tulokset ovat erittäin mielenkiintoisia joidenkin stabiilien lentoratojen löydyttyä niin, että leija voi hakeutua niille ilman mitään palautetta ja jatkaa lentämistä niillä kunnes olosuhteet muuttuvat. Nämä stabiilit lentoradat ovat käyttökelpoisia makaavan kahdeksikon muotoisina ja mahdollisesti niitä voidaan käyttää leijavoimaloissa leijan ohjaamiseksi helpommin luonnollisen stabiilisuuden avulla.

PREFACE

This master of science thesis is based on the work done in the kite generator project at the Department of Mathematics of Tampere University of Technology. The founder of this project and also the examiner of this thesis has been Risto Silvennoinen, but due to health issues he had to be replaced by Keijo Ruohonen as the examiner in the beginning of the writing process. Risto Silvennoinen has been leading the kite generator project and giving ideas about what to research, and Keijo Ruohonen has given useful comments about the contents and the structure of the thesis together with helping to spot some typing errors.

The basics of the theory are based on the calculations done by Ivan Argatov in the summer of 2007 while he was working on the kite generator project. These calculations concentrated on estimating the power of a kite generator and were released in [1] and later in another form revised by me as [13]. Then Ivan Argatov was working again on the kite generator project in the summer of 2010 together with me when we developed the simulation presented in this thesis. Most of the theory used in the simulation was derived by Ivan Argatov whereas I wrote all the MATLAB code to implement the formulas derived by Ivan Argatov. Together we created a working simulator rather quickly and I presented some simulation results first at the AWEC 2011 conference `http://www.awec2011.com/pauli-rautakorpi-tampere-u-o-t/`. Then Ivan Argatov wrote an internal report about the simulation with the theory used in the simulation and some details about how the formulas were derived. This thesis is then meant to be a detailed documentation of our simulator so that I'll derive all the theory again with more detail and also document all the implementation details. In most cases I've followed Argatov's ideas when deriving the formulas used in the simulation, but some parts like the aerodynamic force using the lift tilt angle β and deriving the tether deformation model are different.

Tampere, 19 November 2013

Pauli Rautakorpi

CONTENTS

NOMENCLATURE

$\langle\ \rangle$	Time average for one orbit of kite
α	Angle between vectors $\hat{\mathbf{w}}_\mathbf{p}$ and $\hat{\mathbf{w}}$ like shown in 3.2
α_k	Kite angle of attack
α_{k0}	Kite angle of attack in overhead position
β	Lift tilt control angle of kite
η	Direction angle of initial velocity
θ	Angle between vertical and kite position vector like shown in 3.1
$\dot{\theta}$	Time derivative of angle θ
θ_τ	Derivative of angle θ with respect to dimensionless time τ
$\theta_{\tau\tau}$	Second derivative of angle θ with respect to dimensionless time τ
ξ	x-coordinate of some point in tether used as integration variable
ρ	Dimensionless length of kite position vector $\rho = \frac{r}{l_0}$
ρ_τ	Derivative of radial coordinate ρ with respect to dimensionless time τ
$\rho_{\tau\tau}$	Second derivative of radial coordinate ρ with respect to dimensionless time τ
ρ_a	Density of air
ρ_l	Density of tether material
τ	Dimensionless time $\tau = \frac{V_0}{l_0}t$
ϕ	Angle between wind direction and kite position vector in horizontal plane like shown in 3.1
$\dot{\phi}$	Time derivative of angle ϕ
ϕ_τ	Derivative of angle ϕ with respect to dimensionless time τ
$\phi_{\tau\tau}$	Second derivative of angle ϕ with respect to dimensionless time τ
ψ	Tether length difference control angle of kite
A	Effective wing area of kite
A^{line}	Cross section area of tether
a_i	Even Fourier coefficients of control function
b_i	Odd Fourier coefficients of control function
$C_\parallel$	Normal reaction coefficient that is drag coefficient of tether or cylinder perpendicular to airflow
$C_\perp$	Longitudinal reaction coefficient that is friction coefficient for airflow along tether
C_D	Drag coefficient of kite
C_L	Lift coefficient of kite

D	Scalar component of kite aerodynamic force along airflow called drag
d	Distance between two tether attachment points in kite
d^{line}	Tether diameter
E	Tensile modulus of tether material
E_g	Generated energy
$\hat{\mathbf{e}}_\theta$	Unit vector of local coordinate system at kite towards increasing θ like defined in (3.3)
$\hat{\mathbf{e}}_\phi$	Unit vector of local coordinate system at kite towards increasing ϕ like defined in (3.3)
$\hat{\mathbf{e}}_\mathbf{r}$	Unit vector of local coordinate system at kite in radial direction like defined in (3.3)
F	Shorthand notation for $F = F_{wp}^{\text{gra}} + F_{wp}^{\text{line}}$
$\mathbf{F^{aer}}$	Aerodynamic force acting on kite
$\mathbf{F^{gra}}$	Gravity of kite
F_{wp}^{gra}	Scalar component of kite gravity along vector $\hat{\mathbf{w}}_\mathbf{p}$ defined as $F_{wp}^{\text{gra}} = \mathbf{F^{gra}} \cdot \hat{\mathbf{w}}_\mathbf{p}$
$\mathbf{F}_\perp^{\mathbf{laer}}$	Component of aerodynamic force acting on tether perpendicular to kite position vector
$F_\perp^{\text{laer}}$	Scalar component of aerodynamic force acting on tether perpendicular to kite position vector
$f_\perp^{\text{laer}}(x)$	Shorthand notation for $f_\perp^{\text{laer}}(x) = \frac{1}{2}\rho_a d^{\text{line}} C_\perp \frac{x^2}{r^2} \lVert \mathbf{W}_\mathbf{e}^\mathbf{p} \rVert^2$
$\mathbf{F}_\mathbf{fric}^\mathbf{line}$	Tension component at kite perpendicular to position vector caused by tether drag
$F_{\text{fric}}^{\text{line}}$	Scalar component of tension at kite perpendicular to position vector caused by tether drag
F_{wp}^{line}	Scalar component of tension at kite perpendicular to position vector and along $\hat{\mathbf{w}}_\mathbf{p}$
$\mathbf{F^{tot}}$	Resultant of all forces acting on kite
F_θ^{tot}	Sum of all force components acting on kite along $\hat{\mathbf{e}}_\theta$
F_ϕ^{tot}	Sum of all force components acting on kite along $\hat{\mathbf{e}}_\phi$
F_r^{tot}	Sum of all force components acting on kite along $\hat{\mathbf{e}}_\mathbf{r}$
G_e	Aerodynamic efficiency of kite defined in (3.68)
g	Acceleration due to gravity
$\hat{\mathbf{i}}$	Unit vector of Cartesian coordinate system, downwind
$\hat{\mathbf{j}}$	Unit vector of Cartesian coordinate system
$\hat{\mathbf{k}}$	Unit vector of Cartesian coordinate system, upwards

$\mathbf{L}$	Lift force acting on kite perpendicular to airflow
L	Scalar component of kite aerodynamic force perpendicular to airflow called lift
l	Undeformed tether length
l_0	Starting length of tether
l_r	Real tether length with tensile strain
Δl	Length difference of two tethers
$\mathbf{M}$	Coordinate transformation matrix defined in (3.4)
m	Mass of kite
N	Number of even and odd harmonics in control function
N_l	Number of tethers
P	Mechanical power of kite generator
$\mathbf{r}$	Kite position vector
$\dot{\mathbf{r}}$	Kite velocity vector
$\ddot{\mathbf{r}}$	Kite acceleration vector
$\dot{\mathbf{r}}_\perp$	Projection of kite velocity vector perpendicular to kite position vector $\dot{\mathbf{r}}_\perp = \dot{\mathbf{r}} - (\hat{\mathbf{e}}_\mathbf{r} \cdot \dot{\mathbf{r}})\hat{\mathbf{e}}_\mathbf{r}$
r	Length of kite position vector that is the distance between kite and tether attachment point
$\hat{\mathbf{s}}$	Unit vector perpendicular to $\hat{\mathbf{w}}$ and pointing towards right wingtip when looked from kite
$\hat{\mathbf{s}}_\mathbf{p}$	Unit vector defined as $\hat{\mathbf{s}}_\mathbf{p} = \hat{\mathbf{e}}_\mathbf{r} \times \hat{\mathbf{w}}_\mathbf{p}$
T	Tether tension as scalar
T_p	Period of periodic control function
T_x	Tether tension in longitudinal direction of x coordinate
T_y	Tether tension in transverse direction of y coordinate
$\mathbf{t_1}$, $\mathbf{t_2}$, $\mathbf{t_3}$	Basis vectors for definition of control angle β
$\hat{\mathbf{t}}_\perp$	Direction vector for kite movement perpendicular to kite position vector
V	Wind speed as scalar
V_0	Wind speed at reference altitude
$V_\parallel$	Scalar component of wind velocity along kite position vector $V_\parallel = \hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}$
V_L	Scalar component of kite velocity along kite position vector $V_L = \hat{\mathbf{e}}_\mathbf{r} \cdot \dot{\mathbf{r}}$
$\mathbf{v}_\mathrm{rel}$	Kite velocity relative to air around it $\mathbf{v}_\mathrm{rel} = \dot{\mathbf{r}} - \mathbf{W} = -\mathbf{W}_\mathbf{e}$
$v_\perp$	Kite speed in direction perpendicular to kite position vector
$\mathbf{W}$	Wind velocity vector
$\mathbf{W}_\parallel$	Component of wind velocity along kite position vector $\mathbf{W}_\parallel = (\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W})\hat{\mathbf{e}}_\mathbf{r}$

$\mathbf{W}_\perp$	Component of wind velocity perpendicular to kite position vector $\mathbf{W}_\perp = \mathbf{W} - \mathbf{W}_\parallel$
$\mathbf{W_e}$	Effective wind vector defined as $\mathbf{W_e} = \mathbf{W} - \dot{\mathbf{r}}$ that is airflow around kite
$\mathbf{W}_\mathbf{e}^\parallel(x)$	Component of effective wind around tether along kite position vector as function of longitudinal coordinate x
$\mathbf{W}_\mathbf{e}^\perp(x)$	Component of effective wind around tether perpendicular to kite position vector as function of longitudinal coordinate x
$\mathbf{W}_\mathbf{e}^\mathbf{p}$	Component of effective wind perpendicular to kite position vector $\mathbf{W}_\mathbf{e}^\mathbf{p} = \mathbf{W_e} - (\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W_e})\hat{\mathbf{e}}_\mathbf{r}$
$\hat{\mathbf{w}}$	Unit vector along effective wind and airflow at kite
$\hat{\mathbf{w}}_\mathbf{p}$	Unit vector along vector $\mathbf{W}_\mathbf{e}^\mathbf{p}$
x	Longitudinal coordinate of tether along kite position vector that is zero at support point and grows to r at kite
y	Transverse coordinate of tether

1. INTRODUCTION

Current wind power plant technology can reach and extract only a small part of the wind energy. Wind turbines with a rotor and a generator on top of a tower can't be significantly larger, higher and thus more powerful because of the limits of structural integrity. Thus, contemporary wind turbine technology can only extract energy from winds close to the earth and most of the energy literally flows above the power plants. Additionally, the aerodynamic friction of earth's surface slows down the airflow near the surface.

For example, clouds moving quickly over the wind turbines clearly show how much renewable and emissionless energy is available in the winds blowing freely out of reach for current wind turbines. Naturally, extracting energy from these strong winds at high altitude has been attempted with many different methods, but so far none of them has become commercially feasible because of implementation difficulties. Like many other ideas, using kites for energy production popped up during the oil crisis in the end of the 70's. However, when oil prices decreased again, the research of kite wind generators stopped quickly and there were only some publications. The most significant of those publications [11] is also the basis for this work. It introduces the basic idea of the kite power plant type presented in the next chapter and also some results about kite speed and power that are simplified versions of those derived in [1] and [13]. Thus, Loyd presented the interesting possibility to extract energy from high altitude winds effectively with a large and fast-flying kite.

In the 21th century some research groups are again interested in the idea of extracting energy from high altitude winds with kites. This idea seems now so promising that these groups have been researching continuously for many years with excitement, and also several newly-found companies have started developing and testing kite power plant prototypes, check for example `http://www.energykitesystems.net/` for links. Nowadays there are lighter and stronger kite fabric and line materials available. The increasing popularity of paragliders and power kites as hobby equipment has made kites more common. Modern electronics and computer technology have made real-time optimization and automatic remote control a lot easier and cheaper. These and other changes in conditions and new enabling technologies have now made kite wind generators maybe the most promising concept for extracting energy from high altitude winds. There are already several working small-scale pro-

totypes using power kites that have demonstrated the feasibility of the concept at least in small scale. From these prototypes the KiteGen prototype presented in publications [3] and [4] has a similar working principle as presented in the next chapter and they also present some estimations of power from computer simulations.

I've done some research on such pumping kite generators together with Ivan Argatov. As the first task Ivan Argatov derived formulas to estimate the power of a kite generator, and these calculations were released in [1] and later in another form revised by me as [13]. Then after deriving estimations of the speed of the kite and the mechanical power using analytical methods the goal of our research has been to study the behavior of such system with more detailed simulation. Thus, we developed a simulator together to simulate the energy generation phase of a pumping kite generator by building the required theory on the basics we had in our power calculations. Ivan Argatov worked on the theoretical methods used in the simulation whereas I wrote all the code to implement the formulas derived by Ivan Argatov. Together we created a working simulator rather quickly and I presented some simulation results based on my tests at the AWEC 2011 conference `http://www.awec2011.com/pauli-rautakorpi-tampere-u-o-t/`. The main purpose of this thesis is to be a complete documentation of our simulation covering all the details of both the theory and the simulator implementation. Many areas of the theory needed for this simulation are already presented in our previous publications like the forces acting on the kite and the kite speed law. However, I've also included detailed descriptions of those to present all the required theory in one place together with all the new ideas related to tether modeling, for example.

In the next chapter I'll introduce the concept of a pumping kite generator to present the working principle and provide the necessary information to understand the approximations and assumptions done in the theory chapter. Then in the third chapter I'll derive all the formulas to present a detailed description of all the theory needed for the simulation. I'll first define the coordinate systems used and then present the equations of motion before the aerodynamic force acting on the kite. Then I'll present two different options for the control angle of the kite used in previous publications and derive the equations needed to calculate the kite aerodynamic force in both cases. Then I'll present the aerodynamic force acting on the tether and model its effect on the effective tether length together with elasticity. In the end of the theory chapter I'll derive formulas for calculating proper initial conditions including the kite speed law.

The fourth chapter is a detailed description of the simulation implementation with all the interesting details and tricks used in the attached example code. I'll present function hierarchy and structure first before detailed descriptions of all functions to document all the implementation details. Then the fifth chapter shows some

simulation results given by the attached simulator code. The first simulation tests with constant tether length give some very interesting results including stabilization of the kite on some stable orbits with pre-defined sinusoidal periodic control.

2. WORKING PRINCIPLE OF A PUMPING KITE GENERATOR

A kite generator is a wind power plant that produces electricity and extracts energy from wind by flying a kite. Thus, when talking about kite generators, it is assumed that the mechanical energy extracted from wind is converted to electrical energy in the generator. Of course, it is also possible to use the mechanical energy extracted from wind by a kite in a more direct way to pump water or pull a ship, for example, but here the focus is in generating electricity. However, this chapter is about kite flight and mechanical power, and the electrical properties of the generator are not considered here. Thus, it is possible to use these results to estimate the mechanical power of any system using energy extracted from wind by a kite as long as the assumptions made about the kite and the tether are valid.

In this chapter the pumping type of a kite generator shown in Figure 2.1 is considered with only one kite connected to the ground station with just one tether. It is thus assumed that the generator is on the ground in the ground station and the kite is remotely controlled with some control systems placed in the kite instead of several tethers. It is also assumed that the kite is always flown downwind in the high power zone of the wind window. The wind window means the part of the sky where the kite can fly and the high power zone is in the middle of the wind window around the direction that gives the best possible power not too close to ground and in not too large angle from wind direction. Changing the flying direction of the kite based on wind direction is simple when building the ground station on a rotating platform or placing the tether exit on top of the ground station.

Electricity generation with this type of a kite generator is based on a working cycle with two phases that are energy generation phase and retraction phase. In the energy generation phase the kite pulls the tether with maximal power out of the ground station where the tether is wound on a drum that rotates the generator. In retraction phase the pulling force of the kite is minimized and the tether is pulled back onto the drum using the generator as a motor. In the retraction phase a part of the energy generated in the energy generation phase is lost, but the energy loss in the retraction phase can be minimized by optimizing the structure and the flight path of the kite to produce large and small pulling force alternatively when needed.

In the energy generation phase the pulling force and power of the kite can be

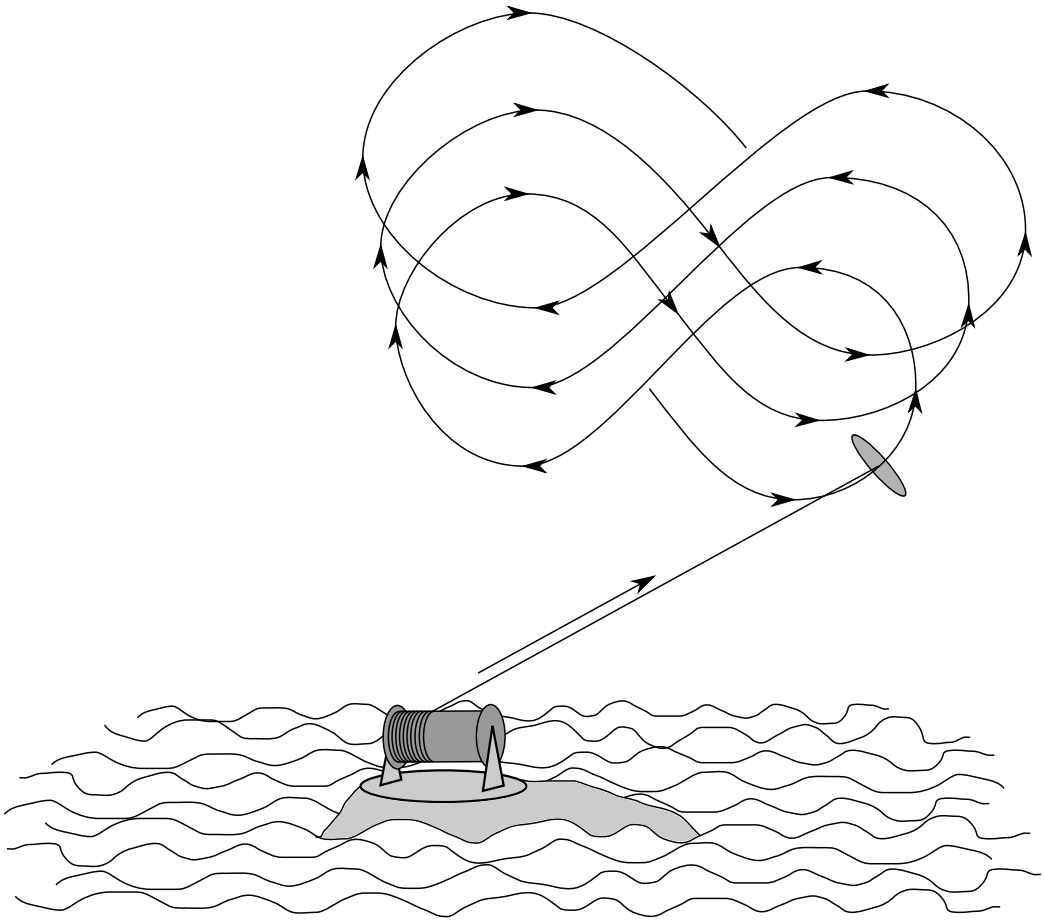

Figure 2.1: Principle of the pumping kite generator considered here during the energy generation phase of the working cycle. The trajectory of the kite is drawn for three lying-eight orbits, but it is possible to make many times more orbits before the retraction phase. The orbit shown is just one typical example because the orbit can be selected freely based on real-time optimization.

maximized by making the kite fly crosswind on lying-eight orbits with a speed of several times the wind speed. The automatic control keeps the kite on the optimal trajectory so that the kite extracts energy effectively from the high altitude wind that is stronger and more steady than at lower altitudes. Thanks to the fast crosswind motion the kite extracts energy from a large cross-section area unlike the cross-section constrained by the rotor diameter of conventional wind turbines. Thanks to the large airspeed the aerodynamic forces acting on the kite are strong and thus produce a large tension to the tether that rotates the generator. In this phase the lift-to-drag ratio of the kite is maximized so that the kite flies fast and generates the maximal power. Based on this working principle it is assumed here that the aerodynamic lift of the kite is at least some times bigger than the aerodynamic drag and the kite is assumed to fly in the high power zone of the wind window in the downwind direction. The shape of the orbit is not so important for the methods used here to estimate the power, but in reality the lying-eight orbit is the simplest possible

orbit that doesn't twist the tether and should work well. The type of the kite is not important either since kite properties are modeled here with some parameters describing the important properties of the kite.

In the retraction phase one option to reduce the pulling force of the kite effectively is to maneuver the kite to an equilibrium position above the ground station for the time needed to reel in the tether. Then the airspeed of the almost stationary kite is close to the wind speed and thus the aerodynamic forces acting on the kite are significantly weaker than in the energy generation phase. The almost vertical tether drops back into the ground station partly assisted by the gravity and thus the energy loss is minimal when the tether is reeled back in. However, there are so many factors affecting the energy loss and duration of the retraction phase so that it is not really possible to consider the retraction phase in a general power estimation without taking into account the details of the construction. Thus, the retraction phase is not included in the power estimation in this chapter to preserve generality and the power formula gives the average mechanical power for one of the orbits in the energy generation phase since conditions and power change when the tether is longer for the following orbits.

In this pumping kite generator the trajectory of the kite can be continuously optimized for changing conditions so that the electricity generation remains effective and the kite stays in air in controlled flight. As basic adjustments the flying direction of the kite is changed according to wind direction and also flight altitude can be adjusted by changing tether length and angle of inclination so that the kite remains in the highest power zone of the wind window in changing wind conditions. Of course, it is also possible to optimize the size and shape of orbits for different conditions. The variation of tether length and thus the duration of energy generation and retraction phases is also adjustable, so it is easily possible to synchronize several pumping kite generators so that one reels in the kite while others generate energy and thus the overall electricity generation is continuous. Thanks to all these adjustable properties it is possible to optimize the electricity generation of a pumping kite generator in very different conditions with a sophisticated real-time optimization and control system.

3. THEORY OF OUR KITE SIMULATOR

3.1 Coordinate systems

Assuming that the vertical component of the wind speed is zero we place a right-handed Cartesian coordinate system to the kite generator so that x-axis is always along the wind direction (downwind) and z-axis upwards. Denoting the unit vectors of this Cartesian coordinate system for x-, y- and z-axis as $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$, respectively, the wind vector $\mathbf{W}$ is

$$\mathbf{W} = V\hat{\mathbf{i}}, \tag{3.1}$$

where the wind speed V is a scalar. The wind speed V usually depends on altitude, and many models for this vertical wind gradient exist.

We'll also use the spherical coordinates r, θ and ϕ defined so that the position vector of the kite $\mathbf{r}$ is in the Cartesian coordinates

$$\mathbf{r} = r\sin\theta\cos\phi\,\hat{\mathbf{i}} + r\sin\theta\sin\phi\,\hat{\mathbf{j}} + r\cos\theta\,\hat{\mathbf{k}}. \tag{3.2}$$

In this definition r is the distance between the kite and the origin that is the attachment point of the tether in the ground station. Then θ is the angle between vertical and the position vector, and ϕ is the angle between the downwind direction and the kite in the horizontal plane and is positive counterclockwise when looking from the ground station.

In this spherical coordinate system we get the local unit vectors

$$\begin{aligned} \hat{\mathbf{e}}_{\mathbf{r}} &= \sin\theta\cos\phi\,\hat{\mathbf{i}} + \sin\theta\sin\phi\,\hat{\mathbf{j}} + \cos\theta\,\hat{\mathbf{k}}, \\ \hat{\mathbf{e}}_{\theta} &= \cos\theta\cos\phi\,\hat{\mathbf{i}} + \cos\theta\sin\phi\,\hat{\mathbf{j}} - \sin\theta\,\hat{\mathbf{k}} \quad \text{and} \\ \hat{\mathbf{e}}_{\phi} &= -\sin\phi\,\hat{\mathbf{i}} + \cos\phi\,\hat{\mathbf{j}} \end{aligned} \tag{3.3}$$

by differentiating the position vector (3.2) with respect to each spherical coordinate and normalizing. The local unit vectors $\hat{\mathbf{e}}_{\mathbf{r}}$, $\hat{\mathbf{e}}_{\theta}$ and $\hat{\mathbf{e}}_{\phi}$ are orthonormal and form a right-handed coordinate system to the kite position. The coordinate transformation matrix

$$\mathbf{M} = \begin{pmatrix} \sin\theta\cos\phi & \cos\theta\cos\phi & -\sin\phi \\ \sin\theta\sin\phi & \cos\theta\sin\phi & \cos\phi \\ \cos\theta & -\sin\theta & 0 \end{pmatrix} \tag{3.4}$$

representing these local unit vectors in the Cartesian coordinates is thus orthogonal and its inverse is $\mathbf{M}^{-1} = \mathbf{M}^\top$.

Now the wind vector (3.1) represented in the basis formed by the vectors $\hat{\mathbf{e}}_\mathbf{r}$, $\hat{\mathbf{e}}_\theta$ and $\hat{\mathbf{e}}_\phi$ is

$$\mathbf{W} = \mathbf{M}^\top \begin{pmatrix} V \\ 0 \\ 0 \end{pmatrix} = V \begin{pmatrix} \sin\theta\cos\phi \\ \cos\theta\cos\phi \\ -\sin\phi \end{pmatrix} = V\Big(\sin\theta\cos\phi\,\hat{\mathbf{e}}_\mathbf{r} + \cos\theta\cos\phi\,\hat{\mathbf{e}}_\theta - \sin\phi\,\hat{\mathbf{e}}_\phi\Big). \tag{3.5}$$

The gravity of the kite $\mathbf{F}^\mathbf{gra} = -mg\,\hat{\mathbf{k}}$ is similarly

$$\mathbf{F}^\mathbf{gra} = \mathbf{M}^\top \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} = -mg \begin{pmatrix} \cos\theta \\ -\sin\theta \\ 0 \end{pmatrix} = -mg\Big(\cos\theta\,\hat{\mathbf{e}}_\mathbf{r} - \sin\theta\,\hat{\mathbf{e}}_\theta\Big), \tag{3.6}$$

where g is the acceleration due to gravity as scalar and m is the mass of the kite.

In this spherical coordinate system the position vector $\mathbf{r}$ of the kite is simply

$$\mathbf{r} = r\,\hat{\mathbf{e}}_\mathbf{r},$$

and the velocity vector of the kite is

$$\dot{\mathbf{r}} = \dot{r}\,\hat{\mathbf{e}}_\mathbf{r} + r\dot{\theta}\,\hat{\mathbf{e}}_\theta + r\dot{\phi}\sin\theta\,\hat{\mathbf{e}}_\phi, \tag{3.7}$$

where differentiation with respect to time is marked with dots. Similarly, the acceleration vector of the kite represented in the local basis of the spherical coordinate system is

$$\begin{aligned} \ddot{\mathbf{r}} \;=\; & \big(\ddot{r} - r\dot{\theta}^2 - r\dot{\phi}^2\sin^2\theta\big)\hat{\mathbf{e}}_\mathbf{r} + \big(2\dot{r}\dot{\theta} + r\ddot{\theta} - r\dot{\phi}^2\sin\theta\cos\theta\big)\hat{\mathbf{e}}_\theta \\ & + \big(2\dot{r}\dot{\phi}\sin\theta + 2r\dot{\theta}\dot{\phi}\cos\theta + r\ddot{\phi}\sin\theta\big)\hat{\mathbf{e}}_\phi. \end{aligned} \tag{3.8}$$

3.2 Equations of motion

Now we let $\mathbf{F}^\mathbf{tot}$ be the resultant of all real forces acting on the kite or some other body with mass m, and then the Newton's second law of motion is

$$\mathbf{F}^\mathbf{tot} = m\ddot{\mathbf{r}}.$$

With the resultant of forces $\mathbf{F}^\mathbf{tot}$ divided into components

$$\mathbf{F}^\mathbf{tot} = F_r^\mathrm{tot}\,\hat{\mathbf{e}}_\mathbf{r} + F_\theta^\mathrm{tot}\,\hat{\mathbf{e}}_\theta + F_\phi^\mathrm{tot}\,\hat{\mathbf{e}}_\phi$$

and using the component representation of the acceleration vector (3.8) we can divide the Newton's second law into three component equations along the local unit vectors of the spherical coordinate system

$$\begin{aligned} F_r^{\text{tot}} &= m\big(\ddot{r} - r\dot{\theta}^2 - r\dot{\phi}^2\sin^2\theta\big), \\ F_\theta^{\text{tot}} &= m\big(2\dot{r}\dot{\theta} + r\ddot{\theta} - r\dot{\phi}^2\sin\theta\cos\theta\big) \quad \text{and} \\ F_\phi^{\text{tot}} &= m\big(2\dot{r}\dot{\phi}\sin\theta + 2r\dot{\theta}\dot{\phi}\cos\theta + r\ddot{\phi}\sin\theta\big). \end{aligned} \tag{3.9}$$

These equations of motion could be used to simulate the system, but we want to use dimensionless variables here. The spherical coordinates θ and ϕ are already dimensionless angles, but to complete dimensionless spherical coordinate system we have to introduce dimensionless distance $\rho = \frac{r}{l_0}$ from the origin to the kite, where the reference length l_0 is the starting length of the tether. Then we also have to introduce dimensionless time $\tau = \frac{V_0}{l_0}t$, where V_0 is the wind speed at some reference altitude.

We use subscript τ to denote the differentiation with respect to the dimensionless time τ. Using the chain rule we can represent the first and second derivatives of the dimensionless spherical coordinates ρ, θ and ϕ with respect to the dimensionless time τ

$$\begin{aligned} \rho_\tau &= \tfrac{d\rho}{dr}\tfrac{dr}{dt}\tfrac{dt}{d\tau} = \tfrac{1}{l_0}\dot{r}\tfrac{l_0}{V_0} = \tfrac{\dot{r}}{V_0}, & \rho_{\tau\tau} &= \tfrac{d\rho_\tau}{d\dot{r}}\tfrac{d\dot{r}}{dt}\tfrac{dt}{d\tau} = \tfrac{1}{V_0}\ddot{r}\tfrac{l_0}{V_0} = \tfrac{l_0}{{V_0}^2}\ddot{r}, \\ \theta_\tau &= \tfrac{d\theta}{dt}\tfrac{dt}{d\tau} = \dot{\theta}\tfrac{l_0}{V_0}, & \theta_{\tau\tau} &= \tfrac{d\theta_\tau}{d\dot{\theta}}\tfrac{d\dot{\theta}}{dt}\tfrac{dt}{d\tau} = \tfrac{l_0}{V_0}\ddot{\theta}\tfrac{l_0}{V_0} = \tfrac{{l_0}^2}{{V_0}^2}\ddot{\theta}, \\ \phi_\tau &= \tfrac{d\phi}{dt}\tfrac{dt}{d\tau} = \dot{\phi}\tfrac{l_0}{V_0} \quad \text{and} & \phi_{\tau\tau} &= \tfrac{d\phi_\tau}{d\dot{\phi}}\tfrac{d\dot{\phi}}{dt}\tfrac{dt}{d\tau} = \tfrac{l_0}{V_0}\ddot{\phi}\tfrac{l_0}{V_0} = \tfrac{{l_0}^2}{{V_0}^2}\ddot{\phi}. \end{aligned} \tag{3.10}$$

Then we can substitute the spherical coordinate r and time derivatives of r, θ and ϕ with their dimensionless versions in the equations of motion (3.9) to use the dimensionless variables. After these substitutions and some simplifying we get a new form of the equations of motion

$$\begin{aligned} F_r^{\text{tot}} &= \tfrac{m{V_0}^2}{l_0}\big(\rho_{\tau\tau} - \rho{\theta_\tau}^2 - \rho{\phi_\tau}^2\sin^2\theta\big), \\ F_\theta^{\text{tot}} &= \tfrac{m{V_0}^2}{l_0}\big(\rho\theta_{\tau\tau} + 2\rho_\tau\theta_\tau - \rho{\phi_\tau}^2\sin\theta\cos\theta\big) \quad \text{and} \\ F_\phi^{\text{tot}} &= \tfrac{m{V_0}^2}{l_0}\big(\rho\sin\theta\,\phi_{\tau\tau} + 2\rho_\tau\phi_\tau\sin\theta + 2\rho\theta_\tau\phi_\tau\cos\theta\big) \end{aligned} \tag{3.11}$$

with the dimensionless variables. We can then solve the second derivatives of the dimensionless spherical coordinates

$$\begin{aligned} \rho_{\tau\tau} &= \rho{\theta_\tau}^2 + \rho{\phi_\tau}^2\sin^2\theta + \tfrac{l_0}{m{V_0}^2}F_r^{\text{tot}}, \\ \theta_{\tau\tau} &= {\phi_\tau}^2\sin\theta\cos\theta - \tfrac{2}{\rho}\rho_\tau\theta_\tau + \tfrac{l_0}{m{V_0}^2}\tfrac{F_\theta^{\text{tot}}}{\rho} \quad \text{and} \\ \phi_{\tau\tau} &= -\tfrac{2}{\rho}\rho_\tau\phi_\tau - 2\theta_\tau\phi_\tau\tfrac{\cos\theta}{\sin\theta} + \tfrac{l_0}{m{V_0}^2}\tfrac{F_\phi^{\text{tot}}}{\rho\sin\theta} \end{aligned} \tag{3.12}$$

from the equations of motion (3.11) with dimensionless variables. These are included

in the ODE system used to simulate the kite.

We can also represent the kite velocity vector (3.7) using the dimensionless spherical coordinates and their derivatives with respect to the dimensionless time τ. After substitutions from the definition $\rho = \frac{r}{l_0}$ and (3.10) we get

$$\dot{\mathbf{r}} = V_0\rho_\tau\,\hat{\mathbf{e}}_\mathbf{r} + V_0\rho\theta_\tau\,\hat{\mathbf{e}}_\theta + V_0\rho\phi_\tau\sin\theta\,\hat{\mathbf{e}}_\phi.$$

This gives us the dimensionless kite velocity vector

$$\frac{\dot{\mathbf{r}}}{V_0} = \rho_\tau\,\hat{\mathbf{e}}_\mathbf{r} + \rho\theta_\tau\,\hat{\mathbf{e}}_\theta + \rho\phi_\tau\sin\theta\,\hat{\mathbf{e}}_\phi \tag{3.13}$$

that gives us an easy way to calculate velocities in the simulation.

3.3 Effective wind and aerodynamic force of kite

In addition to the spherical coordinate system used to determine kite position we need a local basis at the kite to determine kite orientation and aerodynamic force. To build the local basis at the kite we first define the effective wind vector

$$\mathbf{W_e} = \mathbf{W} - \dot{\mathbf{r}} \tag{3.14}$$

which is the airflow at the kite and thus determines the aerodynamic forces acting on the kite. Here $\dot{\mathbf{r}}$ is the velocity vector of the kite. Let $\hat{\mathbf{w}}$ be the unit vector along the effective wind vector

$$\hat{\mathbf{w}} = \frac{\mathbf{W_e}}{\|\mathbf{W_e}\|}. \tag{3.15}$$

We also introduce the unit vector $\hat{\mathbf{s}}$, that is perpendicular to the airflow and thus to the vector $\hat{\mathbf{w}}$ and points towards the right wingtip of the kite when looked from the kite so that the straight line through the wingtips of the kite is parallel to the plane determined by vectors $\hat{\mathbf{w}}$ and $\hat{\mathbf{s}}$. Thus, if the kite is like an airplane with the fuselage aligned with the airflow, then $\hat{\mathbf{w}}$ points backwards in the direction of the airflow, $\hat{\mathbf{s}}$ from the left wingtip towards the right wingtip and $\hat{\mathbf{w}} \times \hat{\mathbf{s}}$ upwards in the direction of lift when looking from the kite. Unit vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$, $\hat{\mathbf{e}}_\mathbf{r}$, $\hat{\mathbf{w}}$ and $\hat{\mathbf{s}}$ together with angles θ and ϕ are illustrated in Figure 3.1.

We then divide the aerodynamic force acting on the kite to three orthogonal components that are lift, drag and side force just like it is common for airplanes. From these the drag D is the component along the airflow and lift L is the component perpendicular to the airflow and to the straight line through the wingtips by definition. Thus, the drag is along the vector $\hat{\mathbf{w}}$ and the direction of the lift is defined by the vector $\hat{\mathbf{w}} \times \hat{\mathbf{s}}$. The side force is along the vector $\hat{\mathbf{s}}$, but we assume that the kite has some kind of tail or stable form to keep it aligned with the airflow well enough.

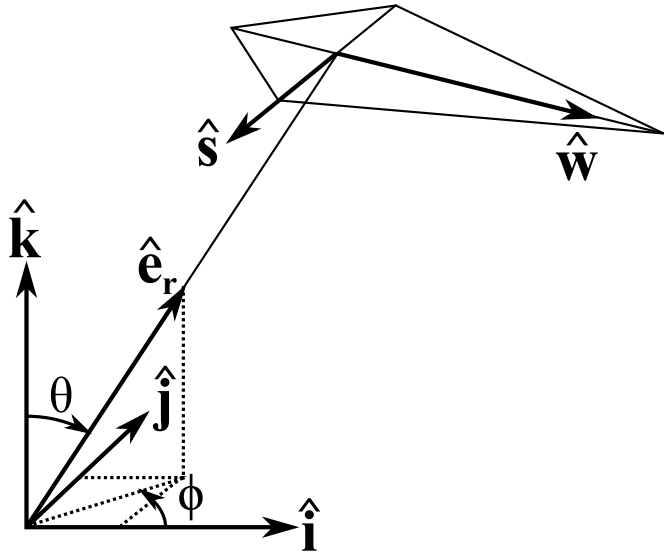

Figure 3.1: An illustration of unit vectors and angles. The kite is aligned with the airflow and we are looking at the underside of the kite.

Thanks to the assumed good directional stability the side force should stay pretty small so that it is not significant for our simulation. Thus, we don't model yaw movements of the kite and can neglect the side force. Using these we can present the aerodynamic force $\mathbf{F^{aer}}$ acting on the kite as

$$\mathbf{F^{aer}} = L(\hat{\mathbf{w}} \times \hat{\mathbf{s}}) + D\,\hat{\mathbf{w}}, \tag{3.16}$$

where the lift and drag have the commonly used formulas

$$L = \frac{1}{2}\rho_a A C_L \|\mathbf{W_e}\|^2 \quad \text{and} \quad D = \frac{1}{2}\rho_a A C_D \|\mathbf{W_e}\|^2, \tag{3.17}$$

that are, for example, presented in [14]. Here ρ_a is the air density, A the characteristic wing area of the kite, C_L the lift coefficient of the kite and C_D the drag coefficient of the kite. The lift and drag coefficients change as a function of the angle of attack of the kite, and this function depends on the shape of the wing and defines the aerodynamics model of the kite.

In the analysis we look at the effective wind and kite movement in the plane spanned by the local basis vectors $\hat{\mathbf{e}}_\theta$ and $\hat{\mathbf{e}}_\phi$, so we now define some vectors in this plane that we need later. Let $\mathbf{W_e^p}$ be the projection of the effective wind vector $\mathbf{W_e}$ onto the plane spanned by the vectors $\hat{\mathbf{e}}_\theta$ and $\hat{\mathbf{e}}_\phi$, so we have

$$\mathbf{W_e^p} = \mathbf{W_e} - (\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W_e})\hat{\mathbf{e}}_\mathbf{r}, \tag{3.18}$$

and let the unit vector along this vector be

$$\hat{\mathbf{w}}_\mathbf{p} = \frac{\mathbf{W}_\mathbf{e}^\mathbf{p}}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|}. \tag{3.19}$$

Thus, vectors $\hat{\mathbf{e}}_\mathbf{r}$ and $\hat{\mathbf{w}}_\mathbf{p}$ are perpendicular and we can define the third unit vector

$$\hat{\mathbf{s}}_\mathbf{p} = \hat{\mathbf{e}}_\mathbf{r} \times \hat{\mathbf{w}}_\mathbf{p} \tag{3.20}$$

that completes the orthogonal right-handed basis formed by vectors $\hat{\mathbf{e}}_\mathbf{r}$, $\hat{\mathbf{w}}_\mathbf{p}$ and $\hat{\mathbf{s}}_\mathbf{p}$. It is worth to notice that the projection of the vector $\hat{\mathbf{s}}$ onto the plane spanned by the vectors $\hat{\mathbf{e}}_\theta$ and $\hat{\mathbf{e}}_\phi$ is not parallel to the vector $\hat{\mathbf{s}}_\mathbf{p}$ in general case.

3.4 Kite orientation and control angle

In the simulation we control the kite flight path by banking the kite so that the lift force is tilted and gives the side component needed for turning. In other words, we rotate the kite along its roll axis so that the side component of the rotated lift force makes the kite turn. At least two different definitions for kite control angle are used in kite generator research. One definition introduced in [5] and used for example in [1] originates from modeling a dual tether kite controlled by the length difference of the two tethers. Another definition introduced in [15] and used for example in [2] is like the roll angle of an airplane. Both of these control angle options are implemented in the simulation, so we'll introduce them here as ψ and β, respectively, and derive representations of the kite aerodynamic force using them. These two control angles are slightly different ways to define the roll control of the kite and are suitable for modeling slightly different control systems but have only small value differences. Thus, either ψ or β is enough to define the roll control of the kite.

3.4.1 Tether length difference control angle ψ

The control angle ψ is defined in [5] as the bank angle caused by the length difference Δl of two tethers with the distance d between their attachment points in the kite. Thus, in this definition we simply have

$$\psi = \arcsin \frac{\Delta l}{d}. \tag{3.21}$$

Just like originally in [5] we'll assume that the tethers are straight and thus in the direction of $\hat{\mathbf{e}}_\mathbf{r}$ when defining the kite orientation for different values of control angle ψ. We'll also have to assume like in [5] that the line between the two tether attachment points in the kite is parallel to the unit vector $\hat{\mathbf{s}}$ and perpendicular to

the effective wind direction $\hat{\mathbf{w}}$, so we assume symmetric placement of the tether attachment points together with the earlier assumption of good directional stability of the kite.

With these assumptions we have the length difference Δl in the direction of $\hat{\mathbf{e}}_\mathbf{r}$ and the distance d between attachment points in the direction of $\hat{\mathbf{s}}$. The unit vector $\hat{\mathbf{s}}$ points from the left wingtip towards the right wingtip when looking from the kite, so the vector from the left attachment point the right attachment point is $d\hat{\mathbf{s}}$ and the projection of this vector to the tether direction is Δl. We define Δl to be positive when the right wingtip is farther from the ground station. Thus, we have $\Delta l = d\hat{\mathbf{s}} \cdot \hat{\mathbf{e}}_\mathbf{r}$ and

$$\hat{\mathbf{s}} \cdot \hat{\mathbf{e}}_\mathbf{r} = \frac{\Delta l}{d} = \sin\psi. \tag{3.22}$$

We want to know the components of the unit vector $\hat{\mathbf{s}}$ in the basis formed by unit vectors $\hat{\mathbf{e}}_\mathbf{r}$, $\hat{\mathbf{w}}_\mathbf{p}$ and $\hat{\mathbf{s}}_\mathbf{p}$ since we need it to get the direction of lift. In (3.22) we already have the component of $\hat{\mathbf{s}}$ in the direction of $\hat{\mathbf{e}}_\mathbf{r}$, and we temporarily denote the two still unknown components with a_w and a_s so that we have

$$\hat{\mathbf{s}} = \sin\psi\, \hat{\mathbf{e}}_\mathbf{r} + a_w \hat{\mathbf{w}}_\mathbf{p} + a_s \hat{\mathbf{s}}_\mathbf{p}. \tag{3.23}$$

We can get a_w using the fact that $\hat{\mathbf{s}}$ is perpendicular to the effective wind direction $\hat{\mathbf{w}}$ by definition. We can use the definition of $\hat{\mathbf{w}}$ (3.15) and the component representation of $\hat{\mathbf{s}}$ (3.23) to get

$$\hat{\mathbf{w}} \cdot \hat{\mathbf{s}} = \frac{\mathbf{W}_\mathbf{e}}{\|\mathbf{W}_\mathbf{e}\|} \cdot \left(\sin\psi\, \hat{\mathbf{e}}_\mathbf{r} + a_w \hat{\mathbf{w}}_\mathbf{p} + a_s \hat{\mathbf{s}}_\mathbf{p}\right).$$

Then we can use the definitions of $\mathbf{W}_\mathbf{e}^\mathbf{p}$ (3.18) and $\hat{\mathbf{w}}_\mathbf{p}$ (3.19) to get

$$\hat{\mathbf{w}} \cdot \hat{\mathbf{s}} = \frac{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|\hat{\mathbf{w}}_\mathbf{p} + (\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e})\hat{\mathbf{e}}_\mathbf{r}}{\|\mathbf{W}_\mathbf{e}\|} \cdot \left(\sin\psi\, \hat{\mathbf{e}}_\mathbf{r} + a_w \hat{\mathbf{w}}_\mathbf{p} + a_s \hat{\mathbf{s}}_\mathbf{p}\right),$$

and calculating the dot product gives

$$\hat{\mathbf{w}} \cdot \hat{\mathbf{s}} = \frac{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|}{\|\mathbf{W}_\mathbf{e}\|} a_w + \frac{\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e}}{\|\mathbf{W}_\mathbf{e}\|} \sin\psi.$$

By definition we have $\hat{\mathbf{w}} \cdot \hat{\mathbf{s}} = 0$ and thus we get

$$a_w = -\frac{\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e}}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|} \sin\psi. \tag{3.24}$$

Then only one of the three components of $\hat{\mathbf{s}}$ is still unknown, and we can get it from normalization of the unit vector $\hat{\mathbf{s}}$. We have $\|\hat{\mathbf{s}}\| = 1$ and thus we get the

equation

$$\sin^2\psi + {a_w}^2 + {a_s}^2 = 1$$

from (3.23). We can solve this equation for the only unknown a_s and get

$$a_s = \pm\sqrt{1 - \sin^2\psi - {a_w}^2} = \pm\sqrt{\cos^2\psi - {a_w}^2}.$$

We have to choose the root with the plus sign here because $\hat{\mathbf{s}}$ and $\hat{\mathbf{s}}_\mathbf{p}$ are roughly in the same direction instead of opposite directions with small control angle due to their definitions. Thus, after substituting a_w from (3.24) we get

$$a_s = \sqrt{\cos^2\psi - \frac{(\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W_e})^2}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|^2}\sin^2\psi}. \tag{3.25}$$

With this we can get the component form of $\hat{\mathbf{s}}$ we wanted by substituting (3.24) and (3.25) into (3.23). Thus, we get

$$\hat{\mathbf{s}} = \sin\psi\,\hat{\mathbf{e}}_\mathbf{r} - \frac{\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W_e}}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|}\sin\psi\,\hat{\mathbf{w}}_\mathbf{p} + \sqrt{\cos^2\psi - \frac{(\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W_e})^2}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|^2}\sin^2\psi}\,\hat{\mathbf{s}}_\mathbf{p} \tag{3.26}$$

just like in [1] and we can use this to calculate the components of the kite aerodynamic force in (3.16).

3.4.2 Lift tilt control angle β

The other control angle definition in [15] uses different basis vectors than introduced here. Their first basis vector $\mathbf{t_1}$ is in the direction of the kite position vector and thus equal to $\hat{\mathbf{e}}_\mathbf{r}$ in our basis. Then two other basis vectors $\mathbf{t_2}$ and $\mathbf{t_3}$ are defined as

$$\mathbf{t_2} = \frac{\mathbf{t_1} \times \mathbf{v}_\text{rel}}{\|\mathbf{t_1} \times \mathbf{v}_\text{rel}\|} \quad \text{and} \quad \mathbf{t_3} = \frac{\mathbf{v}_\text{rel} \times \mathbf{t_2}}{\|\mathbf{v}_\text{rel} \times \mathbf{t_2}\|}, \tag{3.27}$$

where $\mathbf{v}_\text{rel} = \dot{\mathbf{r}} - \mathbf{W} = -\mathbf{W_e}$ is the velocity of the kite relative to the air around it. Then the lift tilt angle β is defined so that the aerodynamic lift force $\mathbf{L}$ acting on the kite is

$$\mathbf{L} = \frac{1}{2}\rho_a A C_L \|\mathbf{v}_\text{rel}\|^2 \left(\sin\beta\,\mathbf{t_2} + \cos\beta\,\mathbf{t_3}\right), \tag{3.28}$$

and the aerodynamic drag force is always in the direction of $-\mathbf{v}_\text{rel}$. This definition already includes all the information needed to calculate the components of the kite aerodynamic force with different values of the lift tilt angle β.

Anyway, for clarity we want to represent the basis vectors $\mathbf{t_2}$ and $\mathbf{t_3}$ using the basis vectors we defined earlier and use in other sections because that gives us a clearer way to calculate the lift direction $\hat{\mathbf{w}} \times \hat{\mathbf{s}}$ using the lift tilt angle β. We can

represent the basis vector $\mathbf{t_2}$ easily using our markings because we have $\mathbf{t_1} = \hat{\mathbf{e}}_\mathbf{r}$ and $\mathbf{v}_{\text{rel}} = -\mathbf{W_e}$, and thus we get

$$\mathbf{t_2} = \frac{\mathbf{t_1} \times \mathbf{v}_{\text{rel}}}{\|\mathbf{t_1} \times \mathbf{v}_{\text{rel}}\|} = \frac{\hat{\mathbf{e}}_\mathbf{r} \times (-\mathbf{W_e})}{\|\hat{\mathbf{e}}_\mathbf{r} \times (-\mathbf{W_e})\|} = -\frac{\hat{\mathbf{e}}_\mathbf{r} \times \mathbf{W_e}}{\|\hat{\mathbf{e}}_\mathbf{r} \times \mathbf{W_e}\|}.$$

We can get even simpler representation for $\mathbf{t_2}$ by noticing that the component of the effective wind vector $\mathbf{W_e}$ along the kite position vector $\hat{\mathbf{e}}_\mathbf{r}$ doesn't matter in the cross product, so we have $\hat{\mathbf{e}}_\mathbf{r} \times \mathbf{W_e} = \hat{\mathbf{e}}_\mathbf{r} \times \mathbf{W_e^p}$ and we get

$$\mathbf{t_2} = -\frac{\hat{\mathbf{e}}_\mathbf{r} \times \mathbf{W_e^p}}{\|\hat{\mathbf{e}}_\mathbf{r} \times \mathbf{W_e^p}\|} = -\frac{\hat{\mathbf{e}}_\mathbf{r} \times \|\mathbf{W_e^p}\|\hat{\mathbf{w}}_\mathbf{p}}{\|\hat{\mathbf{e}}_\mathbf{r} \times \|\mathbf{W_e^p}\|\hat{\mathbf{w}}_\mathbf{p}\|} = -\frac{\hat{\mathbf{e}}_\mathbf{r} \times \hat{\mathbf{w}}_\mathbf{p}}{\|\hat{\mathbf{e}}_\mathbf{r} \times \hat{\mathbf{w}}_\mathbf{p}\|}$$

using the definition (3.19) of $\hat{\mathbf{w}}_\mathbf{p}$. Now vectors $\hat{\mathbf{e}}_\mathbf{r}$ and $\hat{\mathbf{w}}_\mathbf{p}$ are perpendicular unit vectors due to their definitions, so the norm of their cross product is just one and we have already defined $\hat{\mathbf{s}}_\mathbf{p} = \hat{\mathbf{e}}_\mathbf{r} \times \hat{\mathbf{w}}_\mathbf{p}$ in (3.20). Thus, we have simply

$$\mathbf{t_2} = -\hat{\mathbf{e}}_\mathbf{r} \times \hat{\mathbf{w}}_\mathbf{p} = -\hat{\mathbf{s}}_\mathbf{p}. \tag{3.29}$$

Then we can use (3.29) to represent $\mathbf{t_3}$ in the form

$$\mathbf{t_3} = \frac{\mathbf{v}_{\text{rel}} \times \mathbf{t_2}}{\|\mathbf{v}_{\text{rel}} \times \mathbf{t_2}\|} = \frac{(-\mathbf{W_e}) \times (-\hat{\mathbf{s}}_\mathbf{p})}{\|(-\mathbf{W_e}) \times (-\hat{\mathbf{s}}_\mathbf{p})\|} = \frac{\mathbf{W_e} \times \hat{\mathbf{s}}_\mathbf{p}}{\|\mathbf{W_e} \times \hat{\mathbf{s}}_\mathbf{p}\|}.$$

We can then use the unit vector $\hat{\mathbf{w}}$ defined in (3.15) to get

$$\mathbf{t_3} = \frac{(\|\mathbf{W_e}\|\hat{\mathbf{w}}) \times \hat{\mathbf{s}}_\mathbf{p}}{\|(\|\mathbf{W_e}\|\hat{\mathbf{w}}) \times \hat{\mathbf{s}}_\mathbf{p}\|} = \frac{\hat{\mathbf{w}} \times \hat{\mathbf{s}}_\mathbf{p}}{\|\hat{\mathbf{w}} \times \hat{\mathbf{s}}_\mathbf{p}\|}. \tag{3.30}$$

Thus, we now have simple formulas to get the basis vectors $\mathbf{t_2}$ and $\mathbf{t_3}$ so that we can calculate the lift force as

$$\mathbf{L} = \frac{1}{2}\rho_a A C_L \|\mathbf{W_e}\|^2 \left(\cos\beta \, \frac{\hat{\mathbf{w}} \times \hat{\mathbf{s}}_\mathbf{p}}{\|\hat{\mathbf{w}} \times \hat{\mathbf{s}}_\mathbf{p}\|} - \sin\beta \, \hat{\mathbf{s}}_\mathbf{p}\right) \tag{3.31}$$

and the drag force in the direction of $\hat{\mathbf{w}}$ like before so that we have

$$\mathbf{F^{aer}} = L\left(\cos\beta \, \frac{\hat{\mathbf{w}} \times \hat{\mathbf{s}}_\mathbf{p}}{\|\hat{\mathbf{w}} \times \hat{\mathbf{s}}_\mathbf{p}\|} - \sin\beta \, \hat{\mathbf{s}}_\mathbf{p}\right) + D\,\hat{\mathbf{w}} \tag{3.32}$$

with the lift L and drag D defined in (3.17).

3.5 Aerodynamic force of tether

We then look at the aerodynamic forces acting on the tether to get useful models for them and also for the force $\mathbf{F_{fric}^{line}}$ acting on the kite due to them so that we can

simulate both kite and tether dynamics. In this section we approximate the tether with a straight rigid rod that coincides with the kite position vector $\mathbf{r}$ and has a circular cross-section with diameter d^{line}. Thus, the length of the position vector r is also the tether length and we use $x \in [0, r]$ as the longitudinal coordinate along the tether. We divide the wind vector $\mathbf{W} = V\hat{\mathbf{i}}$ in the definition (3.1) into a component along the tether $\mathbf{W}_{\parallel} = (\hat{\mathbf{e}}_{\mathbf{r}} \cdot \mathbf{W})\hat{\mathbf{e}}_{\mathbf{r}}$ and a component perpendicular to the tether $\mathbf{W}_{\perp} = \mathbf{W} - \mathbf{W}_{\parallel}$, that according to (3.5) are

$$\mathbf{W}_{\parallel} = V \sin\theta \cos\phi \, \hat{\mathbf{e}}_{\mathbf{r}} \quad \text{and} \quad \mathbf{W}_{\perp} = V\big(\cos\theta \cos\phi \, \hat{\mathbf{e}}_{\theta} - \sin\phi \, \hat{\mathbf{e}}_{\phi}\big). \tag{3.33}$$

Now we get the aerodynamic normal reaction force $d\mathbf{F}_{\perp}^{\mathbf{laer}}$ acting on a tether piece with length dx using the formula

$$d\mathbf{F}_{\perp}^{\mathbf{laer}} = \frac{1}{2}\rho_a d^{\text{line}} C_{\perp} \|\mathbf{W}_{\mathbf{e}}^{\perp}(x)\| \mathbf{W}_{\mathbf{e}}^{\perp}(x) dx \tag{3.34}$$

that is introduced in the book [9] and used for kite tether in [8]. In other words, this normal reaction force is the component of the aerodynamic force perpendicular to the tether. Here $C_{\perp}$ is the drag coefficient of a long cylinder perpendicular to the airflow and $\mathbf{W}_{\mathbf{e}}^{\perp}(x)$ is a transverse projection of the effective wind at the piece of the tether. The effective wind projection $\mathbf{W}_{\mathbf{e}}^{\perp}$ is a function of the longitudinal coordinate x, at the kite we have $\mathbf{W}_{\mathbf{e}}^{\perp}(r) = \mathbf{W}_{\mathbf{e}}^{\mathbf{p}}$ and at the ground station only the contribution of wind is present $\mathbf{W}_{\mathbf{e}}^{\perp}(0) = \mathbf{W}_{\perp}$ since the tether can't have any transverse movement at its attachment point. Between these two extremes the part due to the kite movement in the effective wind increases linearly, so similarly to the definition (3.14) we get the transverse effective wind component around the tether

$$\mathbf{W}_{\mathbf{e}}^{\perp}(x) = \mathbf{W}_{\perp} - \frac{x}{r}\dot{\mathbf{r}}_{\perp}, \tag{3.35}$$

where $\dot{\mathbf{r}}_{\perp} = \dot{\mathbf{r}} - (\hat{\mathbf{e}}_{\mathbf{r}} \cdot \dot{\mathbf{r}})\hat{\mathbf{e}}_{\mathbf{r}}$ is the transverse projection of the kite velocity vector. For these equations it is worth to notice the connection

$$\mathbf{W}_{\mathbf{e}}^{\mathbf{p}} = \mathbf{W}_{\mathbf{e}}^{\perp}(r) \overset{(3.35)}{=} \mathbf{W}_{\perp} - \dot{\mathbf{r}}_{\perp}, \tag{3.36}$$

that gives a useful form of (3.35)

$$\mathbf{W}_{\mathbf{e}}^{\perp}(x) = \left(1 - \frac{x}{r}\right)\mathbf{W}_{\perp} + \frac{x}{r}\mathbf{W}_{\mathbf{e}}^{\mathbf{p}}. \tag{3.37}$$

We can get the tether drag force $\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}}$ acting on the kite by looking at a torque balance where the torque of the kite $r\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}}$ has to be equal to the torque of the tether drag forces acting on the whole tether with respect to the tether attachment point on the ground. Thus, we can get the torque due to the tether drag by integrating the

torque of a piece of the tether $xd\mathbf{F}_{\perp}^{\mathbf{laer}}$ with respect to the longitudinal coordinate x over the whole tether. Now this integral and the torque balance give us the force $\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}}$ in the form

$$\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}} = \frac{1}{r}\int_0^r xd\mathbf{F}_{\perp}^{\mathbf{laer}}. \tag{3.38}$$

Substituting the aerodynamic normal reaction force (3.34) into the integral (3.38) for the force $\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}}$ gives us

$$\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}} = \frac{1}{r}\int_0^r \frac{1}{2}\rho_a d^{\text{line}} C_{\perp} \|\mathbf{W}_{\mathbf{e}}^{\perp}(x)\| \mathbf{W}_{\mathbf{e}}^{\perp}(x) x\, dx. \tag{3.39}$$

This formula gives us the tether drag force $\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}}$ at the kite for our simulation.

In the energy generation phase of an effectively working kite generator the kite flies many times faster than the wind speed in the high power zone of the wind window with the tether far from perpendicular to the wind, so $\|\mathbf{W}_{\mathbf{e}}^{\mathbf{p}}\|$ is many times larger than $\|\mathbf{W}_{\perp}\|$ in reality. Thus, the $\mathbf{W}_{\perp}$ term in (3.37) is significant only for small values of the coordinate x close to the ground station and quickly becomes insignificant at larger values of x. Because the integral in (3.38) has the part close to the kite most significant we can use the approximation

$$\mathbf{W}_{\mathbf{e}}^{\perp}(x) \approx \frac{x}{r}\mathbf{W}_{\mathbf{e}}^{\mathbf{p}}, \tag{3.40}$$

with the error diminishing when x increases. Now using the approximation (3.40) together with (3.34) we get

$$d\mathbf{F}_{\perp}^{\mathbf{laer}}(x) \approx \frac{1}{2}\rho_a d^{\text{line}} C_{\perp} \|\frac{x}{r}\mathbf{W}_{\mathbf{e}}^{\mathbf{p}}\| \frac{x}{r}\mathbf{W}_{\mathbf{e}}^{\mathbf{p}} dx = \frac{1}{2}\rho_a d^{\text{line}} C_{\perp} \frac{x^2}{r^2} \|\mathbf{W}_{\mathbf{e}}^{\mathbf{p}}\| \mathbf{W}_{\mathbf{e}}^{\mathbf{p}} dx. \tag{3.41}$$

When the wind doesn't depend on the position like in our definition (3.1) we can now easily calculate the integral in (3.38) and get the result

$$\mathbf{F}_{\mathbf{fric}}^{\mathbf{line}} \approx \frac{1}{r}\int_0^r \frac{1}{2}\rho_a d^{\text{line}} C_{\perp} \frac{x^3}{r^2} \|\mathbf{W}_{\mathbf{e}}^{\mathbf{p}}\| \mathbf{W}_{\mathbf{e}}^{\mathbf{p}} dx = \frac{1}{8}\rho_a d^{\text{line}} r C_{\perp} \|\mathbf{W}_{\mathbf{e}}^{\mathbf{p}}\| \mathbf{W}_{\mathbf{e}}^{\mathbf{p}}. \tag{3.42}$$

This formula can be used in simulation to give a simple approximation for the tether drag.

3.6 Tether deformation model

3.6.1 Tether length changes due to elasticity and deformation

We'll then try to model how tether elasticity and deformation affect the real radial coordinate of the kite. According to the definition of the tensile modulus E we get

the real length of the tether

$$l_r = l + \frac{Tl}{A^{\text{line}}E} = l\left(1 + \frac{T}{A^{\text{line}}E}\right)$$

where l is the undeformed length of the tether and $A^{\text{line}} = \frac{\pi}{4}{d^{\text{line}}}^2$ is the cross-section area of the tether. Thus, for an infinitesimal part of the tether we have

$$dl_r = dl\left(1 + \frac{T}{A^{\text{line}}E}\right). \tag{3.43}$$

In addition to the elasticity the tether is also not straight, so we define y as the position coordinate of the tether part with respect to the position vector of the kite in direction perpendicular to the position vector. For detailed analysis we can use two coordinates y_θ and y_ϕ in the directions of the local position vectors of our radial coordinate system, but as a simple approximation we'll use only the approximate aerodynamic force (3.41) as the load force deforming the tether. Thus, the load is assumed to be only in the direction of $\mathbf{W_e^p}$, so the deformation is also in that direction only and we'll define y to be positive in the direction of $\mathbf{W_e^p}$. With these coordinates we have for the length of the tether part an equation

$${dl_r}^2 = dx^2 + \left(\frac{dy}{dx}dx\right)^2$$

and thus we get

$$dl_r = \sqrt{1 + \left(\frac{dy}{dx}\right)^2}dx. \tag{3.44}$$

Using equations (3.43) and (3.44) we can get the undeformed length dl of the tether part as

$$dl = \frac{1}{1 + \frac{T}{A^{\text{line}}E}}dl_r = \frac{\sqrt{1 + \left(\frac{dy}{dx}\right)^2}}{1 + \frac{T}{A^{\text{line}}E}}dx.$$

Then we get the undeformed length l of the whole tether by integrating along the tether

$$l = \int_0^r dl = \int_0^r \frac{\sqrt{1 + \left(\frac{dy}{dx}\right)^2}}{1 + \frac{T(x)}{A^{\text{line}}E}}dx \tag{3.45}$$

noting that the tension T of the tether changes along the tether and is thus a function of the longitudinal coordinate x.

The tether of a kite generator should be tightly stretched during the normal operation and the tensile stress should be within the material strength limits, so we can assume that the stretched length of the tether isn't a lot larger than undeformed

length and the shape of the tether is quite close to straight. Thus, we'll assume here that we have

$$\frac{T(x)}{A^{\text{line}}E} \ll 1 \quad \text{and} \quad \left|\frac{dy}{dx}\right| \ll 1 \quad \text{for} \quad x \in [0, r]. \tag{3.46}$$

Using the assumptions (3.46) we can then approximate the integrand of (3.45)

$$\frac{\sqrt{1+\left(\frac{dy}{dx}\right)^2}}{1+\frac{T(x)}{A^{\text{line}}E}} \approx \frac{1+\frac{1}{2}\left(\frac{dy}{dx}\right)^2}{1+\frac{T(x)}{A^{\text{line}}E}} = \frac{\left(1+\frac{1}{2}\left(\frac{dy}{dx}\right)^2\right)\left(1-\frac{T(x)}{A^{\text{line}}E}\right)}{1-\left(\frac{T(x)}{A^{\text{line}}E}\right)^2}$$

using the series expansion of square root. Then we can neglect the square of the small number $\frac{T(x)}{A^{\text{line}}E} \ll 1$ in the denominator to get

$$\begin{aligned}\frac{\left(1+\frac{1}{2}\left(\frac{dy}{dx}\right)^2\right)\left(1-\frac{T(x)}{A^{\text{line}}E}\right)}{1-\left(\frac{T(x)}{A^{\text{line}}E}\right)^2} &\approx \left(1+\frac{1}{2}\left(\frac{dy}{dx}\right)^2\right)\left(1-\frac{T(x)}{A^{\text{line}}E}\right) \\ &= 1+\frac{1}{2}\left(\frac{dy}{dx}\right)^2 - \frac{T(x)}{A^{\text{line}}E} - \frac{1}{2}\left(\frac{dy}{dx}\right)^2\frac{T(x)}{A^{\text{line}}E}.\end{aligned}$$

Then the last term is less significant than the others as a product of two small numbers and can thus be neglected. Thus, we get an useful approximation for the integrand of (3.45)

$$\frac{\sqrt{1+\left(\frac{dy}{dx}\right)^2}}{1+\frac{T(x)}{A^{\text{line}}E}} \approx 1+\frac{1}{2}\left(\frac{dy}{dx}\right)^2 - \frac{T(x)}{A^{\text{line}}E}. \tag{3.47}$$

3.6.2 Approximate solution for tether shape

We then should get some approximate solution for tether shape so that we know $\frac{dy}{dx}$ in the tether length integral (3.45) with approximation (3.47). Here we assume that the tether is tightly stretched so that the kite generates a large tension to the tether compared to the inertia effects and other forces acting on the tether, and thus the tether shape is close to straight. We have already defined y coordinate to be positive in the direction of $\mathbf{W}_\mathbf{e}^\mathbf{p}$ after assuming that all the transverse load on the tether is in that direction. We use two scalar components for the tether tension T so that we have T_x in the longitudinal direction of x coordinate and T_y in the transverse direction of y coordinate, both positive in the directions of those coordinate axis. We neglect the gravity of the tether here and thus use only the aerodynamic force acting on the tether in the approximated form (3.41) as the transverse load force.

With these definitions we have

$$\frac{dy}{dx} = \frac{T_y}{T_x}. \tag{3.48}$$

because the tension is along the tether, and this gives us

$$T_y = T_x \frac{dy}{dx}.$$

Then the longitudinal tension T_x is almost as large as the total tension T and almost constant due to the assumption that the tether tension generated by the kite is large compared to other forces acting on the tether. Thus, assuming that $T_x \approx T$ is constant along the tether we can differentiate both sides with respect to x and get

$$\frac{dT_y}{dx} = T\frac{d^2y}{dx^2}. \tag{3.49}$$

When we neglect the inertia we get another equation from the stationary force balance of an infinitesimal tether piece in transverse direction

$$T_y(x) + dT_y(x) - T_y(x) + dF_\perp^{\text{laer}}(x) = 0, \tag{3.50}$$

where we have

$$dF_\perp^{\text{laer}}(x) \approx \frac{1}{2}\rho_a d^{\text{line}} C_\perp \frac{x^2}{r^2} \|\mathbf{W_e^P}\|^2 dx = f_\perp^{\text{laer}}(x)dx \tag{3.51}$$

from (3.41) with $f_\perp^{\text{laer}}(x)$ defined to be equal to $\frac{1}{2}\rho_a d^{\text{line}} C_\perp \frac{x^2}{r^2}\|\mathbf{W_e^P}\|^2$ to shorten the equations. Thus, from (3.50) and (3.51) we have

$$\frac{dT_y}{dx} = -f_\perp^{\text{laer}}(x).$$

This together with the other equation (3.49) gives us

$$T\frac{d^2y}{dx^2} = -f_\perp^{\text{laer}}(x)$$

so that we get

$$\frac{d^2y}{dx^2} = -\frac{f_\perp^{\text{laer}}(x)}{T}.$$

We can then solve this ODE by integrating both sides with respect to x along the tether from ground up to some point ξ to get

$$\int_0^\xi \frac{d^2y}{dx^2}dx = -\int_0^\xi \frac{f_\perp^{\text{laer}}(x)}{T}dx,$$

and by calculating the integral on the left hand side we get

$$\frac{dy(\xi)}{dx} - \frac{dy(0)}{dx} = -\int_0^\xi \frac{f_\perp^{\text{laer}}(x)}{T}dx. \tag{3.52}$$

We don't know the initial value $\frac{dy(0)}{dx}$ in (3.52) yet, but we know as boundary values that the transverse coordinate y has to be zero at both ends because the tether is attached there. To use those boundary values we'll have to integrate both sides of (3.52) again with respect to ξ along the whole tether so that we get

$$\int_0^r \frac{dy(\xi)}{dx}d\xi - \int_0^r \frac{dy(0)}{dx}d\xi = -\int_0^r \int_0^\xi \frac{f_\perp^{\text{laer}}(x)}{T}dxd\xi.$$

By calculating the integrals on the left hand side we then get

$$y(r) - y(0) - r\frac{dy(0)}{dx} = -\int_0^r \int_0^\xi \frac{f_\perp^{\text{laer}}(x)}{T}dxd\xi,$$

so that we can use the boundary values of y to get the initial value we wanted

$$\frac{dy(0)}{dx} = \frac{1}{r}\int_0^r \int_0^\xi \frac{f_\perp^{\text{laer}}(x)}{T}dxd\xi. \tag{3.53}$$

We can then use the initial value (3.53) in the solution (3.52) to get an approximate solution for the tether shape

$$\frac{dy}{dx} = \frac{1}{r}\int_0^r \int_0^\xi \frac{f_\perp^{\text{laer}}(x)}{T}dxd\xi - \int_0^x \frac{f_\perp^{\text{laer}}(x)}{T}dx. \tag{3.54}$$

Then the next step is to substitute $f_\perp^{\text{laer}}(x)$ defined in (3.51) into (3.54) to get

$$\frac{dy}{dx} = \frac{1}{r}\int_0^r \int_0^\xi \frac{1}{2T}\rho_a d^{\text{line}} C_\perp \frac{x^2}{r^2}\|\mathbf{W_e^p}\|^2 dxd\xi - \int_0^x \frac{1}{2T}\rho_a d^{\text{line}} C_\perp \frac{x^2}{r^2}\|\mathbf{W_e^p}\|^2 dx$$

and start calculating the integrals. After calculating the inner integrals we have

$$\frac{dy}{dx} = \frac{1}{r}\int_0^r \frac{1}{6T}\rho_a d^{\text{line}} C_\perp \frac{\xi^3}{r^2}\|\mathbf{W_e^p}\|^2 d\xi - \frac{1}{6T}\rho_a d^{\text{line}} C_\perp \frac{x^3}{r^2}\|\mathbf{W_e^p}\|^2,$$

and calculating the remaining integral gives us

$$\frac{dy}{dx} = \frac{\rho_a d^{\text{line}} C_\perp}{24T} r\|\mathbf{W_e^p}\|^2 - \frac{\rho_a d^{\text{line}} C_\perp}{6T}\frac{x^3}{r^2}\|\mathbf{W_e^p}\|^2.$$

We can then simplify this to get the approximate solution for the tether shape

$$\frac{dy}{dx} = \frac{\rho_a d^{\text{line}} C_\perp}{6T}\|\mathbf{W_e^p}\|^2\left(\frac{r}{4} - \frac{x^3}{r^2}\right) \tag{3.55}$$

that we can use in the tether length integral (3.45) with approximation (3.47).

3.6.3 Connection between radial coordinate of kite and tether length

We can then use the equations we have derived in this tether deformation model section to get one equation that connects the radial coordinate r of the kite to the undeformed tether length l for our simulation. We start building this equation by using approximation (3.47) in the tether length integral (3.45) to get

$$l = \int_0^r \left(1 + \frac{1}{2}\left(\frac{dy}{dx}\right)^2 - \frac{T(x)}{A^{\text{line}}E}\right) dx.$$

Then by neglecting the tension variation along the tether and substituting the approximate solution (3.55) for the tether shape we get

$$l = \int_0^r \left(1 + \frac{1}{2}\left(\frac{\rho_a d^{\text{line}} C_\perp}{6T}\|\mathbf{W_e^p}\|^2\left(\frac{r}{4} - \frac{x^3}{r^2}\right)\right)^2 - \frac{T}{A^{\text{line}}E}\right) dx$$

that can be written for easier integration as

$$l = \left(1 - \frac{T}{A^{\text{line}}E}\right)\int_0^r dx + \frac{1}{2}\left(\frac{\rho_a d^{\text{line}} C_\perp}{6T}\|\mathbf{W_e^p}\|^2\right)^2 \int_0^r \left(\frac{r^2}{16} - \frac{x^3}{2r} + \frac{x^6}{r^4}\right) dx.$$

Calculating the integrals gives us

$$l = \left(1 - \frac{T}{A^{\text{line}}E}\right) r + \frac{1}{2}\left(\frac{\rho_a d^{\text{line}} C_\perp}{6T}\|\mathbf{W_e^p}\|^2\right)^2 \left(\frac{r^3}{16} - \frac{r^3}{8} + \frac{r^3}{7}\right)$$

that we can simplify as

$$l = \left(1 - \frac{T}{A^{\text{line}}E}\right) r + \frac{1}{14}\left(\frac{\rho_a d^{\text{line}} C_\perp}{8T}\|\mathbf{W_e^p}\|^2\right)^2 r^3. \tag{3.56}$$

This equation gives us a connection between the radial coordinate r of the kite and the undeformed tether length l that we need in our simulation.

3.7 Initial conditions

In addition to the equations we'll also need proper initial values for the simulation. We can choose kite position and direction of kite movement as the free initial parameters that we can define for each simulation run. As initial speed of the kite we'll use the value determined by the refined crosswind motion law derived in [1] and also in [13] as the average crosswind motion law. We'll now derive that average crosswind motion law in the following sections.

3.7.1 Geometry of the crosswind motion

In this section we assume that the straight line through the wingtips of the kite is perpendicular to the position vector and thus $\hat{\mathbf{s}} \cdot \hat{\mathbf{e}}_\mathbf{r} = 0$, in other words, the kite isn't banked to turn. In this case the vector $\hat{\mathbf{s}}$ is in the plane spanned by the vectors $\hat{\mathbf{e}}_\theta$ and $\hat{\mathbf{e}}_\phi$, and we also get

$$\begin{aligned}\hat{\mathbf{s}} \cdot \hat{\mathbf{w}}_\mathbf{p} &\overset{(3.19)}{=} \hat{\mathbf{s}} \cdot \frac{\mathbf{W}_\mathbf{e}^\mathbf{p}}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|} \overset{(3.18)}{=} \frac{1}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|}\Big(\hat{\mathbf{s}} \cdot (\mathbf{W}_\mathbf{e} - (\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e})\hat{\mathbf{e}}_\mathbf{r})\Big) \\ &= \frac{1}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|}\Big(\underbrace{(\hat{\mathbf{s}} \cdot \mathbf{W}_\mathbf{e})}_{0} - (\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e})\underbrace{(\hat{\mathbf{s}} \cdot \hat{\mathbf{e}}_\mathbf{r})}_{0}\Big) = 0,\end{aligned}$$

so in this case $\hat{\mathbf{s}} = \hat{\mathbf{s}}_\mathbf{p} \overset{(3.20)}{=} \hat{\mathbf{e}}_\mathbf{r} \times \hat{\mathbf{w}}_\mathbf{p}$. The dot product of the vectors $\hat{\mathbf{s}}$ and $\mathbf{W}_\mathbf{e}$ is zero by definition since the vector $\hat{\mathbf{s}}$ is defined to be perpendicular to the airflow and thus to the vector $\mathbf{W}_\mathbf{e}$.

Thus, vectors $\hat{\mathbf{e}}_\mathbf{r}$, $\hat{\mathbf{w}}$, $\hat{\mathbf{w}}_\mathbf{p}$ and $\hat{\mathbf{w}} \times \hat{\mathbf{s}}$ are all in the same plane in this case. Now let the angle α be the angle between the vectors $\hat{\mathbf{w}}_\mathbf{p}$ and $\hat{\mathbf{w}}$ and we look at the situation in Figure 3.2.

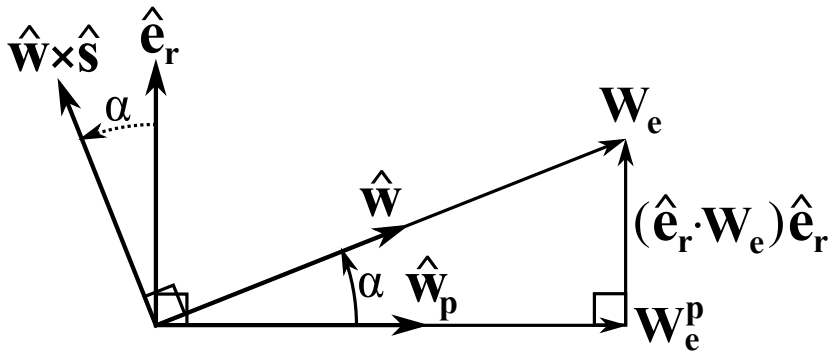

Figure 3.2: An illustration of vectors in the same plane.

We can see in Figure 3.2 useful representations for the sine and cosine of the angle α and get

$$\sin\alpha = \frac{\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e}}{\|\mathbf{W}_\mathbf{e}\|} \quad \text{and} \quad \cos\alpha = \frac{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|}{\|\mathbf{W}_\mathbf{e}\|}. \tag{3.57}$$

We now define components along the position vector $\mathbf{r}$ for wind speed and kite speed so that

$$V_\parallel = \hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W} \quad \text{and} \quad V_L = \hat{\mathbf{e}}_\mathbf{r} \cdot \dot{\mathbf{r}}, \tag{3.58}$$

respectively. With these definitions we can modify (3.57) to the form

$$\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e} \overset{(3.14)}{=} \hat{\mathbf{e}}_\mathbf{r} \cdot (\mathbf{W} - \dot{\mathbf{r}}) \overset{(3.58)}{=} V_\parallel - V_L, \quad \sin\alpha = \frac{V_\parallel - V_L}{\|\mathbf{W}_\mathbf{e}\|} \quad \text{and} \quad \tan\alpha = \frac{V_\parallel - V_L}{\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|}. \tag{3.59}$$

If we assume that the tether is straight, doesn't stretch and is parallel to the kite position vector $\mathbf{r}$, then V_L is the tether unwinding speed in the ground station and $V_\parallel$ is the component of wind speed along the tether.

In Figure 3.2 we can also see that the basis vectors $\hat{\mathbf{w}}$ and $\hat{\mathbf{w}} \times \hat{\mathbf{s}}$ can be represented using vectors $\hat{\mathbf{w}}_\mathbf{p}$ and $\hat{\mathbf{e}}_\mathbf{r}$ as

$$\hat{\mathbf{w}} = \sin\alpha\, \hat{\mathbf{e}}_\mathbf{r} + \cos\alpha\, \hat{\mathbf{w}}_\mathbf{p} \quad \text{and} \quad \hat{\mathbf{w}} \times \hat{\mathbf{s}} = \cos\alpha\, \hat{\mathbf{e}}_\mathbf{r} - \sin\alpha\, \hat{\mathbf{w}}_\mathbf{p}. \tag{3.60}$$

Using these and (3.16) we can also represent the aerodynamic force $\mathbf{F}^\mathbf{aer}$ acting on the kite as

$$\mathbf{F}^\mathbf{aer} = (-L\sin\alpha + D\cos\alpha)\hat{\mathbf{w}}_\mathbf{p} + (L\cos\alpha + D\sin\alpha)\hat{\mathbf{e}}_\mathbf{r}. \tag{3.61}$$

3.7.2 Force balance perpendicular to the position vector

Now we look at the movement of the kite in the direction perpendicular to the position vector and derive a formula for the length of the vector $\mathbf{W}_\mathbf{e}^\mathbf{p}$ from the force balance in the direction of this vector and thus also the vector $\hat{\mathbf{w}}_\mathbf{p}$. Using (3.61) we get the component of the aerodynamic force acting on the kite in the direction of the vector $\hat{\mathbf{w}}_\mathbf{p}$, and the component of the kite gravity $F_{wp}^{\text{gra}} = \mathbf{F}^\mathbf{gra} \cdot \hat{\mathbf{w}}_\mathbf{p}$ acts on the kite in the direction of $\hat{\mathbf{w}}_\mathbf{p}$. The tether is not completely straight in reality and thus the tension at the kite has a component perpendicular to the position vector with a component F_{wp}^{line} in the direction of $\hat{\mathbf{w}}_\mathbf{p}$. All real forces acting on the kite and their components along the vector $\hat{\mathbf{w}}_\mathbf{p}$ are illustrated in Figure 3.3 together with the lift and drag components of the aerodynamic force acting on the kite.

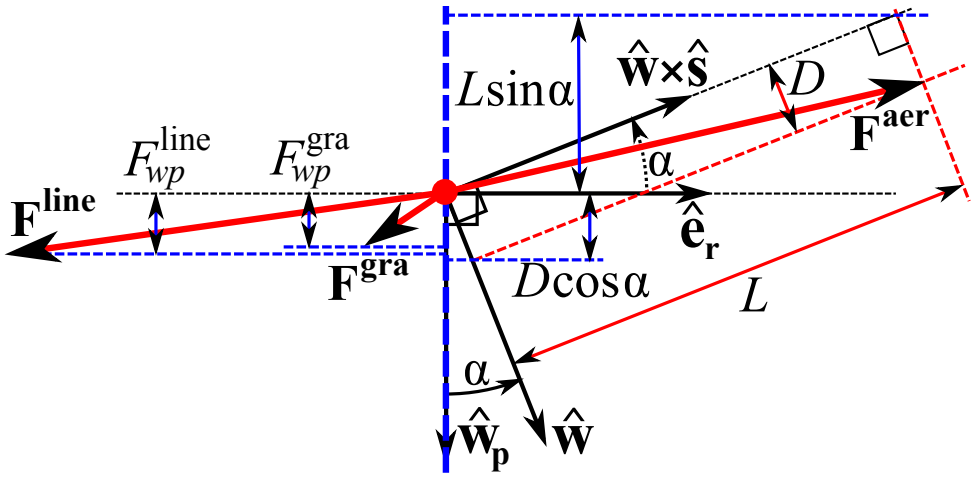

Figure 3.3: Force balance of the kite along the vector $\hat{\mathbf{w}}_\mathbf{p}$. All unit vectors are black, all forces are red together with the lift and drag components, and the direction perpendicular to the position vector together with all components in that direction are blue.

When we neglect the inertia of the kite, the force balance equation in the direction

of the vector $\hat{\mathbf{w}}_\mathbf{p}$ is

$$-L\sin\alpha + D\cos\alpha + F_{wp}^{\text{gra}} + F_{wp}^{\text{line}} = 0. \tag{3.62}$$

We now introduce the shorthand notation

$$F = F_{wp}^{\text{gra}} + F_{wp}^{\text{line}}, \tag{3.63}$$

and according to the force balance (3.62) we have

$$F = L\sin\alpha - D\cos\alpha. \tag{3.64}$$

Now we actually need an equation for $\tan\alpha$ and from (3.64) we get $\tan\alpha = \frac{D}{L} + \frac{F}{L\cos\alpha}$. In an effectively working kite generator the kite lift L should be significantly larger than the kite drag D and also the contribution of other forces F defined in (3.63), so we can assume that the angle α is pretty small with the crosswind speed several times faster than the wind speed. Thus, we can use the approximate equation

$$\tan\alpha \approx \frac{D}{L} + \frac{F}{L}. \tag{3.65}$$

3.7.3 Crosswind motion speed and aerodynamic efficiency

From (3.59) and (3.65) we get the component of the kite airspeed in the direction perpendicular to the position vector

$$\|\mathbf{W}_\mathbf{e}^\mathbf{p}\| \approx (V_\parallel - V_L)\frac{L}{D+F} \overset{(3.63)}{=} (V_\parallel - V_L)\frac{L}{D + F_{wp}^{\text{gra}} + F_{wp}^{\text{line}}}. \tag{3.66}$$

We can call this formula the crosswind motion law since it gives the speed of the crosswind motion of the kite. We already know the other kite airspeed component along the position vector using the wind speed and the tether release speed since in (3.59) we have $\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e} = V_\parallel - V_L$, so with the crosswind motion law (3.66) we know the airspeed of the kite. In the energy generation phase of a kite generator $\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|$ is usually several times the wind speed and thus pretty close to the kite speed when looking at the kite from ground at the ground station. Thus, the kite moves in the direction perpendicular to the tether about $\frac{L}{D+F_{wp}^{\text{gra}}+F_{wp}^{\text{line}}}$ times faster than the effective wind speed along the tether.

However, when we derive a formula for the average power of a kite generator we are interested in the average value of $\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|$. We use angled bracket notation here for values averaged over one lying-eight orbit of the kite or other almost closed orbit, so we are interested in $\langle\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|\rangle$. Since the component of the kite gravity F_{wp}^{gra} in the

crosswind motion law (3.66) accelerates and decelerates the kite in different parts of the orbit as much when the flight altitude doesn't change significantly between orbits we can neglect its effects when looking at the average values over one complete orbit. In addition, we can divide F_{wp}^{line} into two parts, a part $F_{\text{fric}}^{\text{line}}$ caused by aerodynamic forces acting on the tether and a part caused by the gravity of the tether. From these two parts we can neglect the part caused by the gravity when looking at the average values just like with the kite gravity, so as the average value of the crosswind motion law (3.66) we get

$$\langle \|\mathbf{W}_\mathbf{e}^\mathbf{p}\| \rangle \approx \langle V_\parallel - V_L \rangle \frac{L}{D + F_{\text{fric}}^{\text{line}}}. \tag{3.67}$$

Here $F_{\text{fric}}^{\text{line}}$ is the contribution of aerodynamic forces acting on the whole tether in the tether tension component at the kite perpendicular to the position vector. This average crosswind motion law is a really important formula for the estimation of the power of a kite generator.

We now introduce the aerodynamic efficiency G_e of the kite. It determines how many times larger the airspeed component in the direction perpendicular to the position vector tends to be compared to the component along the tether $\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e}$. Thus, with (3.59) we get

$$\langle \|\mathbf{W}_\mathbf{e}^\mathbf{p}\| \rangle = G_e \langle V_\parallel - V_L \rangle, \tag{3.68}$$

where according to the average crosswind motion law (3.67) the aerodynamic efficiency of the kite is

$$G_e \approx \frac{L}{D + F_{\text{fric}}^{\text{line}}}. \tag{3.69}$$

It is worth to notice that in airplane terminology the aerodynamic efficiency G_e of the kite is the glide ratio L/D with a correction due to the additional drag caused by the tether.

3.7.4 Airspeed of the kite

We can now collect the results of previous sections to get useful formulas for the aerodynamic efficiency and airspeed of the kite. We can represent the effective wind vector based on (3.18) as

$$\mathbf{W}_\mathbf{e} = \mathbf{W}_\mathbf{e}^\mathbf{p} + (\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W}_\mathbf{e})\hat{\mathbf{e}}_\mathbf{r}$$

and using (3.19) and (3.59) together with the average crosswind motion law (3.68) we get a useful formula for the average effective wind vector

$$\langle \mathbf{W}_\mathbf{e} \rangle = G_e \langle V_\parallel - V_L \rangle \hat{\mathbf{w}}_\mathbf{p} + \langle V_\parallel - V_L \rangle \hat{\mathbf{e}}_\mathbf{r}. \tag{3.70}$$

We call this formula the kite speed law since it gives the components of the average airspeed of the kite. This means that the airspeed component $\|\mathbf{W_e^p}\|$ perpendicular to the position vector tends to be G_e times larger than the airspeed component $\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W_e} \overset{(3.59)}{=} V_\parallel - V_L$ along the position vector according to the average crosswind motion law (3.68). Because the basis vector $\hat{\mathbf{w}}_\mathbf{p}$ is defined to be perpendicular to the basis vector $\hat{\mathbf{e}}_\mathbf{r}$ we get a simple formula for the average airspeed of the kite

$$\langle \|\mathbf{W_e}\| \rangle = \sqrt{{G_e}^2 + 1} \langle V_\parallel - V_L \rangle. \tag{3.71}$$

When we substitute the force components (3.17) and (3.42) into the aerodynamic efficiency formula (3.69) and use for the tether drag the approximation $\|\mathbf{W_e^p}\| \approx \|\mathbf{W_e}\|$ we get a useful formula for the aerodynamic efficiency of the kite

$$G_e \approx \frac{C_L}{C_D + \dfrac{C_\perp d^{\text{line}} r}{4A}}. \tag{3.72}$$

Here the error of the approximation $\|\mathbf{W_e^p}\| \approx \|\mathbf{W_e}\|$ decreases when the aerodynamic efficiency G_e increases just like the approximations used when deriving the other formulas (3.69) and (3.42) because in the approximation $\|\mathbf{W_e^p}\| \approx \|\mathbf{W_e}\|$ we have actually neglected the constant one inside the square root of (3.71). For an aerodynamically effective kite $\|\mathbf{W_e}\|$ is thus just a bit larger than $\|\mathbf{W_e^p}\|$.

3.7.5 Initial kite speed and tether deformation

We now use $v_\perp$ as the kite speed and $\hat{\mathbf{t}}_\perp$ as the direction vector for kite movement in the plane perpendicular to the kite position vector so that we represent the perpendicular kite velocity as $\dot{\mathbf{r}}_\perp = v_\perp \hat{\mathbf{t}}_\perp$. Then we can start from (3.36) to get a formula for calculating the perpendicular kite speed $v_\perp$ that we need as initial parameter. After taking the norms of both sides of (3.36) and squaring both sides we have

$$\|\mathbf{W_e^p}\|^2 = \|\mathbf{W}_\perp - \dot{\mathbf{r}}_\perp\|^2$$

and we can rewrite the right hand side as

$$\|\mathbf{W}_\perp - \dot{\mathbf{r}}_\perp\|^2 = \left(\mathbf{W}_\perp - v_\perp \hat{\mathbf{t}}_\perp\right) \cdot \left(\mathbf{W}_\perp - v_\perp \hat{\mathbf{t}}_\perp\right) = \|\mathbf{W}_\perp\|^2 - 2\left(\mathbf{W}_\perp \cdot \hat{\mathbf{t}}_\perp\right) v_\perp + {v_\perp}^2.$$

Thus, we have a second order equation

$${v_\perp}^2 - 2\left(\mathbf{W}_\perp \cdot \hat{\mathbf{t}}_\perp\right) v_\perp + \|\mathbf{W}_\perp\|^2 - \|\mathbf{W_e^p}\|^2 = 0$$

for $v_\perp$ and as the solutions we get

$$v_\perp = \left(\mathbf{W}_\perp \cdot \hat{\mathbf{t}}_\perp\right) \pm \sqrt{\left(\mathbf{W}_\perp \cdot \hat{\mathbf{t}}_\perp\right)^2 - \|\mathbf{W}_\perp\|^2 + \|\mathbf{W}_\mathbf{e}^\mathbf{p}\|^2}.$$

Real solutions for $v_\perp$ exist at least when $\|\mathbf{W}_\mathbf{e}^\mathbf{p}\| > \|\mathbf{W}_\perp\|$ that should happen with kite flying properly crosswind in power generation phase of a kite generator. In that case the root with minus sign is negative and our speed $v_\perp$ can't be negative, so the root with plus sign has to be the right root. Thus, we can calculate the perpendicular kite speed $v_\perp$ using the formula

$$v_\perp = \left(\mathbf{W}_\perp \cdot \hat{\mathbf{t}}_\perp\right) + \sqrt{\left(\mathbf{W}_\perp \cdot \hat{\mathbf{t}}_\perp\right)^2 - \|\mathbf{W}_\perp\|^2 + \|\mathbf{W}_\mathbf{e}^\mathbf{p}\|^2}. \tag{3.73}$$

This formula gives us the initial perpendicular kite speed for simulation when we use the average value of $\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|$ given by (3.68) and (3.72) as an approximation that is most accurate in the middle of a lying-eight orbit. The initial direction of kite flight is a free parameter and defines the direction vector $\hat{\mathbf{t}}_\perp$.

The kite speed along the kite position vector is the tether release speed V_L that is also a free parameter in initial conditions just like all three kite position coordinates r, θ and ϕ. Then the initial undeformed length l of the tether can be calculated using (3.56), or alternatively, we can define l as a free parameter and get r using the same equation, but both of these calculations require information about initial tether tension. We can get the initial tether tension using the aerodynamic lift L acting on the kite given by (3.17) as a simple approximation. We can further simplify the calculation by replacing $\|\mathbf{W}_\mathbf{e}\|$ with $\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|$ in the lift formula to get slightly smaller tension approximation

$$T \approx \frac{1}{2}\rho_a A C_L \|\mathbf{W}_\mathbf{e}^\mathbf{p}\|^2$$

to compensate for the fact that some of the lift is in direction perpendicular to the tether to counter drag. Substituting this tension approximation to the tether length equation (3.56) gives us

$$l = \left(1 - \frac{\rho_a A C_L}{2A^{\text{line}}E}\|\mathbf{W}_\mathbf{e}^\mathbf{p}\|^2\right) r + \frac{1}{14}\left(\frac{d^{\text{line}}C_\perp}{4AC_L}\right)^2 r^3 \tag{3.74}$$

as the equation for initial tether length calculations with $\|\mathbf{W}_\mathbf{e}^\mathbf{p}\| = G_e(V_\parallel - V_L)$ from (3.68) and G_e given by (3.69).

4. SIMULATOR IMPLEMENTATION

The energy generation phase of a working kite generator can be simulated using the equations derived in previous sections, and this section describes one way to do so in MATLAB. In this simulation implementation the controls are pre-defined as functions of time and the ODE system is then solved as long as the kite flies in the energy generation phase. The parent function runs the simulation by solving the ODE system (3.12) that is defined in a separate function. Then initial conditions, control function, forces, wind profile and system properties are defined in separate functions. In this section those functions are described starting from the inner functions so that they are known when they're called by other functions. The full version of simulation has variable tether length with the tether unwinding speed as a control together with the control angle, but a simpler version with constant tether length is also presented because it's used for stability test runs in the next section.

4.1 Control definition

The control functions for both control angle and tether release speed are defined as Fourier sums in this example simulation. This gives periodic control functions defined by certain number of Fourier coefficients a_i and b_i so that the control function is

$$\psi = \frac{a_0}{2} + \sum_{i=1}^{N} \left(a_i \cos\left(\frac{2\pi i t}{T_p}\right) + b_i \sin\left(\frac{2\pi i t}{T_p}\right) \right), \tag{4.1}$$

where t is time, T_p is the period of the control function and N is the number of odd and even harmonics. This is implemented in one small function called FourierControlFun that gets time t, period T_p, Fourier coefficients and N as parameters and returns the value of the control function. The Fourier coefficients are given as one vector with first all a_i and then all b_i. All these parameters except the time t are selected for each simulation run to get the desired control functions.

4.2 System and environment properties

All constants that may require editing are collected in one function called Value in this example simulation. They're returned in one switch structure based on their names for simplicity. These constants include effective wing area, angle of attack

in overhead position and mass as parameters of the kite. Then the tether parameters are characteristic length and diameter together with normal and longitudinal reaction coefficients as the aerodynamic properties. The number of tethers is also one adjustable constant just like density and elastic modulus of the tether material. Then average wind speed V_0 at reference altitude, acceleration due to gravity and density of air are adjustable environment parameters.

In addition to these constants some simulation parameters and limits are also defined in the same Value function for easy adjustments. These are the order of control function definition N in (4.1), initial θ and ϕ angles to define the starting position of the kite, safety limit for θ and limit for dimensionless flight time τ together with solver tolerances and step sizes. The safety limit for θ defines the minimum angle above horizon so that the kite is considered crashed and simulation ends when angle θ exceeds its safety limit.

4.2.1 Wind conditions

The wind speed as a function of altitude is modeled in a separate function called WindProfile in this example simulation. This function gets the Cartesian component presentation of the position vector as a parameter together with time and returns the Cartesian component presentation of the wind vector $\mathbf{W}$. As third input parameter is the average wind speed V_0 at some reference altitude to avoid calling the Value function for the definition of V_0 every time the wind vector is calculated. Time and position dependent wind speed and direction are thus possible in this example code with a modified WindProfile function, but this example code has only a simple version of the WindProfile function with the fixed wind direction defined in (3.1), two simple models for wind speed as a function of altitude and possibility to add sinusoidal gusts. The strength and frequency of gusts is defined within the WindProfile function just like the selection of wind gradient law and its parameters.

The two alternative vertical wind gradient laws defining the wind speed as a function of altitude are presented in [7]. Simpler of the two is the wind gradient power law according to Hellmann that is better presented in [10]. It gives the wind speed V at the altitude z as

$$V(z) = V_0 \left(\frac{z}{z_0} \right)^a , \tag{4.2}$$

with the average wind speed V_0 at the reference altitude z_0 and the Hellmann's exponent a. Usually the reference altitude z_0 is 10 m like in this example code. The Hellmann's exponent depends on roughness of the ground, shape of the terrain and temperature gradient, for example, but $1/7$ is used in this example code as a typical value for open flat areas. This simple formula is commonly used for the engineering

of wind turbines although the value of the Hellmann's exponent depends quite a lot of the place and is difficult to estimate.

The other wind gradient law used in this example simulation is the logarithmic law valid for the Prandtl layer in boundary layer flow. It gives the wind speed V at the altitude z as

$$V(z) = V_0 \frac{\ln \frac{z}{z_r}}{\ln \frac{z_0}{z_r}}, \tag{4.3}$$

with the average wind speed V_0 at the reference altitude z_0 and the roughness length z_r. The roughness length depends on the roughness of the surface and varies from 10^{-4} m for water surfaces to 1 m for cities. According to [7] the thickness of the Prandtl layer varies between 10 m and 150 m with the meteorological conditions. The logarithmic law usually underestimates the wind speed at higher altitudes and the power law is usually less accurate, so neither of these wind gradient laws is really valid for typical flight altitudes of the kites in kite generators.

The Cartesian component presentation of the wind vector $\mathbf{W}$ returned by the WindProfile function isn't directly useful in other parts of the simulation because the simulation uses the dimensionless spherical coordinates ρ, θ and ϕ. Thus, a function called WindSpherical is used instead of WindProfile when the wind vector is needed in other parts of the code so that it can do the coordinate transformations between the Cartesian coordinates used for defining the wind vector $\mathbf{W}$ and the spherical coordinate system used elsewhere. It takes the average wind speed V_0 at some reference altitude, the characteristic length l_0 of the tether and the dimensionless spherical coordinates ρ, θ and ϕ together with time as parameters. It then calculates the Cartesian component presentation of the position vector $\mathbf{r}$ using (3.2) with $r = l_0\rho$, calls the WindProfile function to get the wind vector $\mathbf{W}$ and then returns component presentation of the dimensionless wind vector $\frac{\mathbf{W}}{V_0}$ in the local spherical coordinate basis. The conversion from the Cartesian component presentation to component presentation in the local spherical coordinate basis is done by multiplying the inverse $\mathbf{M}^\top$ of the coordinate transformation matrix (3.4) with the Cartesian component presentation of the wind vector $\mathbf{W}$.

4.2.2 Aerodynamics of the kite

The example simulation includes a simple calculation of the angle of attack α_k of the kite to allow modeling the lift and drag coefficients for the kite as functions of the angle of attack. The lift and drag coefficients of the kite are calculated in functions called Cl and Cd, respectively, and those two functions include the aerodynamic model of the kite. In this example code both take just the angle of attack α_k of the kite as the only parameter and return the lift and drag coefficients, but more detailed aerodynamic models may require additional parameters to calculate the Reynolds

number, for example. This example code uses the common simple formulas

$$C_L = C_{L_0} + C_{L_\alpha}\alpha_k \quad \text{and} \quad C_D = C_{D_0} + KC_L{}^2 \tag{4.4}$$

that are also used in [16] to model the aerodynamics of the kite. With these the lift coefficient increases linearly with increasing angle of attack α_k and the drag coefficient is the sum of profile drag coefficient C_{D_0} and induced drag coefficient $KC_L{}^2$. Values for these aerodynamics parameters in the example simulation are $C_{L_0} = 0.3$, $C_{L_\alpha} = 5$, $C_{D_0} = 0.02$ and $K = 0.08$ that give a maximum glide ratio of 12.5 for the wing without tethers at about 2.3 degrees as the optimal angle of attack. Values of these aerodynamic parameters depend on the shape and type of the kite, and with proper parameter values for the kite this simple model should be quite realistic at smaller angles of attack that give good glide ratio. These simple formulas don't even try to model the stall, so they're no longer realistic at all when the angle of attack gets above 10 degrees, but in energy generation phase such situations shouldn't happen because the angle of attack should be kept close to the optimal for glide ratio.

4.3 Forces

4.3.1 Kite aerodynamic force with control angle ψ

The aerodynamic force acting on the kite is calculated in a function called FaerKite when the tether length difference control angle ψ is used. This function gets the control angle ψ, time t and the dimensionless spherical coordinates ρ, θ and ϕ together with their first derivatives with respect to the dimensionless time τ as parameters. To save some calls of the Value function also V_0, l_0 and density of air are given as parameters, but still the Value function is called twice in the beginning to get the effective wing area A of the kite and the angle of attack α_{k0} in overhead position.

Then the dimensionless kite velocity vector $\frac{\dot{\mathbf{r}}}{V_0}$ is calculated using (3.13). The next step is to get the dimensionless effective wind vector $\frac{\mathbf{W_e}}{V_0}$ like in (3.14) by subtracting the dimensionless kite velocity vector from the dimensionless wind vector $\frac{\mathbf{W}}{V_0}$ given by the WindSpherical function. Then the unit vector $\hat{\mathbf{w}}$ defined in (3.15) gives us the direction of the drag force and is calculated by normalizing the dimensionless effective wind vector $\frac{\mathbf{W_e}}{V_0}$.

Then the next step is to calculate the unit vector $\hat{\mathbf{s}}$ needed to get the direction of the lift force in (3.16). This $\hat{\mathbf{s}}$ is given by (3.26), but before that the vectors $\frac{\mathbf{W_e^p}}{V_0}$, $\hat{\mathbf{w}}_\mathbf{p}$ and $\hat{\mathbf{s}}_\mathbf{p}$ are needed. The vector $\frac{\mathbf{W_e^p}}{V_0}$ is calculated by just making the first component $\hat{\mathbf{e}}_\mathbf{r} \cdot \frac{\mathbf{W_e}}{V_0}$ of $\frac{\mathbf{W_e}}{V_0}$ zero like defined in (3.18). Then normalizing it gives $\hat{\mathbf{w}}_\mathbf{p}$ like defined in

(3.19), and $\hat{\mathbf{s}}_\mathbf{p}$ is calculated like in its definition (3.20) to get all the vectors needed in (3.26). Then (3.26) is used to get $\hat{\mathbf{s}}$ using the dimensionless vectors $\frac{\mathbf{W_e}}{V_0}$ and $\frac{\mathbf{W_e^p}}{V_0}$ instead of $\mathbf{W_e}$ and $\mathbf{W_e^p}$ because the difference cancels in both fractions.

Then the next step is to calculate the angle of attack α_k of the kite. The simple angle of attack calculation in this example simulation assumes that the wing stays in a fixed angle from the position vector in the plane defined by the position vector and the effective wind vector. This fixed angle is defined as the angle of attack α_{k0} of the kite in stationary overhead position when the kite stays directly above the ground station with vertical position vector, zero velocity and the unit vector $\hat{\mathbf{w}}$ in the direction of the wind vector. With this assumption the angle of attack α_k of the kite is calculated as

$$\alpha_k = \alpha_{k0} + \arcsin \frac{\hat{\mathbf{e}}_\mathbf{r} \cdot \mathbf{W_e}}{\|\mathbf{W_e}\|} \tag{4.5}$$

from the direction of the effective wind vector compared to the kite position vector. The asin function in MATLAB returns values of arcsin in radians, so the conversion to degrees is needed in code since both α_k and α_{k0} are in degrees. Also the dimensionless vector $\frac{\mathbf{W_e}}{V_0}$ can be used instead of $\mathbf{W_e}$ because the difference cancels.

In reality the kite rotates around its bridle point when the position of the aerodynamic center changes due to differences in the angle of attack so that the tether tension acting on the bridle point and the total aerodynamic force acting on the aerodynamic center almost line up and cancel each other if the kite weight is small. Then the tether is also not straight in reality and this tether sag affects the angle of attack of the kite. However, these two factors affecting the angle of attack are too difficult to model in this simple example simulation, but error in the angle of attack with the simple model is smaller when the tether is closer to straight and the bridle point is farther from the kite compared to the chord of the wing.

With the angle of attack of the kite known the magnitudes of lift and drag are then calculated using (3.17) and functions Cl and Cd to get the lift and drag coefficients for that angle of attack. Here V_0 is needed to get the real magnitude of the effective wind instead of the dimensionless version. Then the aerodynamic force vector is finally returned after calculating it using (3.16). This like all other force vectors in the example simulation is presented as components in the local spherical coordinate basis.

4.3.2 Kite aerodynamic force with control angle β

The aerodynamic force acting on the kite is calculated in a function called FaerKiteB when the lift tilt control angle β is used. This function is otherwise similar to the FaerKite function, but the lift tilt angle β is given as the control angle parameter instead of the tether length difference control angle ψ and the formula (3.32) is used

to calculate the aerodynamic force in the end instead of (3.16). Thus, the vector $\hat{\mathbf{s}}$ is not used in this version and the unit vector $\mathbf{t_3}$ is calculated instead using (3.30) to simplify the final formula.

4.3.3 Approximate tether aerodynamic force

The aerodynamic force acting on the tethers is calculated in a function called FaerTether0 using the simple approximation (3.42). This function gets time t and the dimensionless spherical coordinates ρ, θ and ϕ together with their first derivatives with respect to the dimensionless time τ as parameters. To save some calls of the Value function also V_0, l_0 and density of air are given as parameters, but still the Value function is called thrice in the beginning to get the number of tethers, the diameter d^{line} of one tether and the normal reaction coefficient $C_\perp$.

Then the dimensionless kite velocity vector $\frac{\dot{\mathbf{r}}}{V_0}$, the dimensionless effective wind vector $\frac{\mathbf{W_e}}{V_0}$ and the vector $\frac{\mathbf{W_e^P}}{V_0}$ are calculated just like in the FaerKite function. Then the length of the tethers is just the radial coordinate $r = l_0\rho$ because the tethers are assumed to be straight here. Then everything is ready for calculating the aerodynamic force acting on one tether using (3.42) and returning this force vector in the local spherical coordinate basis after multiplying it with the number of tethers. This force vector is the force at the kite due to the tether aerodynamic force.

4.3.4 Integrated tether aerodynamic force

This example simulation has also another option to calculate the aerodynamic force acting on the tethers with better accuracy. This function called FaerTether1 calculates all three force components by integrating them separately along the tether, so the component along the tether is also calculated unlike in FaerTether0. For compatibility, the parameters are the same as for FaerTether0, and the returned force vector is presented similarly in the local spherical coordinate basis so that these two functions are interchangeable options.

In the beginning the Value function is called five times to get the longitudinal reaction coefficient $C_\parallel$ and the integration tolerance in addition to the number of tethers, the diameter d^{line} of one tether and the normal reaction coefficient $C_\perp$. Then the dimensionless kite velocity vector $\frac{\dot{\mathbf{r}}}{V_0}$ and the length of the tethers $r = l_0\rho$ are calculated just like in FaerTether0 before the three integrations. The integrations are done using the quadl function of MATLAB because it may be more efficient at higher accuracies with smooth integrands with its adaptive Lobatto quadrature according to MATLAB documentation [12]. Use of the quadl quadrature function requires that the integrand is written as a separate vectorized function that accepts

a vector of points and returns a vector of values at those points. Unfortunately, the WindSpherical function is needed inside all three integrand functions for each point to get local wind conditions and the example implementation of WindSpherical doesn't work for vectors, so all the points have to be calculated in slow for loop inside the three integrand functions after creating a vector of zeros for values to reserve memory before the loop.

First of the three components to be calculated is the component of the aerodynamic force parallel to the tether. The component $d\mathbf{F}_{\parallel}^{\mathbf{laer}}$ of the aerodynamic force acting along a tether piece with length dx can be calculated using the formula

$$d\mathbf{F}_{\parallel}^{\mathbf{laer}} = \frac{1}{2}\rho_a d^{\text{line}} C_{\parallel} \|\mathbf{W}_{\mathbf{e}}^{\parallel}(x)\| \mathbf{W}_{\mathbf{e}}^{\parallel}(x) dx \tag{4.6}$$

that is introduced in the book [9] and used for kite tether in [8]. Here $C_{\parallel}$ is the longitudinal reaction coefficient of the tether and $\mathbf{W}_{\mathbf{e}}^{\parallel}(x)$ is the effective wind projection along the piece of the tether. Because the tether is assumed to be straight here, $\mathbf{W}_{\mathbf{e}}^{\parallel}(x) = \mathbf{W}_{\parallel}(x) - V_L \hat{\mathbf{e}}_{\mathbf{r}}$ is in the direction of $\hat{\mathbf{e}}_{\mathbf{r}}$. Thus, the aerodynamic force parallel to the tether is calculated by integrating (4.6) along the tether as

$$\mathbf{F}_{\parallel}^{\mathbf{laer}} = \frac{1}{2}\rho_a d^{\text{line}} C_{\parallel} \int_0^r \|\mathbf{W}_{\mathbf{e}}^{\parallel}(x)\| \mathbf{W}_{\mathbf{e}}^{\parallel}(x) dx. \tag{4.7}$$

with everything else constant in the integrand except the first component $\mathbf{W}_{\parallel}(x)$ of the wind vector in $\mathbf{W}_{\mathbf{e}}^{\parallel}(x)$ changes due to different wind speed at different altitudes. The integrand function called LongInt just calls the WindSpherical function to get the wind vector, calculates scalar value of the effective wind along the tether by subtracting the first component of the kite velocity from the first component of the wind vector, and returns the absolute value of the effective wind multiplied by the effective wind for each point in a loop. The first force component along the tether is then calculated using (4.7) with a call of quadl to calculate the integral along the tether and V_0 included because the integrand function uses dimensionless velocities.

Then the other two components are calculated using (3.39) for the two components separately. In these cases the integrand functions called PerpInt1 and PerpInt2 call the WindSpherical function to get the wind vector, calculate the two last components $\mathbf{W}_{\mathbf{e}}^{\perp}(x)$ of the effective wind using (3.35) and return the value of the integrand function with $x\|\mathbf{W}_{\mathbf{e}}^{\perp}(x)\|$ multiplied by correct component of $\mathbf{W}_{\mathbf{e}}^{\perp}(x)$ for each point in a loop. Then the force components perpendicular to the tether are calculated using (3.39) with a call of quadl to calculate the integral along the tether and V_0 included because the integrand function uses dimensionless velocities. With all three force components calculated for one tether the result is returned after multiplying with the number of tethers.

4.3.5 Gravity

The gravity acting on the kite is calculated in a function called Fgra. The gravity of the kite is given by (3.6), but in this function the gravity of the tethers is also included by adding the effective mass of the tethers to the mass m of the kite. The effective mass of two tethers is calculated to be $\frac{1}{4}\rho_l \pi {d^{\text{line}}}^2 r$ at [6] with the density ρ_l of the tether material, so the effective mass of one tether $\frac{1}{8}\rho_l \pi {d^{\text{line}}}^2 r$ is half of the real mass giving the idea that half of the tether is supported by the kite and half by the ground station.

The Fgra function gets the dimensionless spherical coordinates ρ and θ as parameters together with the characteristic length l_0 of the tether and the mass m of the kite. Then the Value function is called four times to get the acceleration due to gravity g, the number of tethers, the diameter d^{line} of one tether and the density ρ_l of the tether material. The length of the tethers is just the radial coordinate $r = l_0\rho$ with the tethers assumed to be straight, and then the effective mass of one tether $\frac{1}{8}\rho_l \pi {d^{\text{line}}}^2 r$ is calculated. Then the gravity vector is returned using (3.6) with the effective mass of all tethers added to the mass of the kite.

4.3.6 Tether tension

The connection between the radial coordinate r of the kite, the undeformed tether length l and the tether tension T is defined by the equation (3.56) for this example simulation. This connection is implemented in a function called Tension that returns the tension T in the tether when both the radial coordinate r of the kite and the undeformed tether length l are known. This function gets the dimensionless undeformed tether length $\frac{l}{l_0}$, time t and the dimensionless spherical coordinates ρ, θ and ϕ together with their first derivatives with respect to the dimensionless time τ as parameters. To save some calls of the Value function also V_0, l_0 and density of air are given as parameters, but still the Value function is called four times in the beginning to get the number N_l of tethers, the diameter d^{line} of one tether, the normal reaction coefficient $C_\perp$ and the tensile modulus E of the tether material. Then the dimensionless kite velocity vector $\frac{\dot{\mathbf{r}}}{V_0}$, the dimensionless effective wind vector $\frac{\mathbf{W_e}}{V_0}$ and the vector $\frac{\mathbf{W_e^P}}{V_0}$ are calculated just like in the FaerKite function, and everything is ready for using the equation (3.56).

However, the tension T has to be solved first from the equation (3.56) before it can be calculated. To do so the equation (3.56) can be represented as

$$\frac{a}{T^2} - bT = c \quad \text{with} \quad a = \frac{r^3}{14}\left(\frac{\rho_a d^{\text{line}} C_\perp}{8}\|\mathbf{W_e^P}\|^2\right)^2, \quad b = \frac{r}{A^{\text{line}}E} \quad \text{and} \quad c = l - r. \tag{4.8}$$

This represented equation (4.8) can then be solved analytically using Maple to get

$$T = \frac{1}{6}\frac{X^{\frac{2}{3}} + 4c^2 - 2cX^{\frac{1}{3}}}{bX^{\frac{1}{3}}} \quad \text{with} \quad X = 108ab^2 - 8c^3 + 12\sqrt{3}\sqrt{a\,(27ab^2 - 4c^3)}b \quad (4.9)$$

as the only real solution with the assumptions that T, a and b are positive and c is real like necessarily happens in reality.

Then a, b and c in (4.8) are calculated with $r = l_0\rho$ and V_0 included because of the dimensionless velocities used in the code. Also d^{line} is multiplied by the number of tethers in a together with $A^{\text{line}} = N_l\frac{\pi}{4}d^{\text{line}^2}$ in b to take into account the effects of all tethers instead of just one in (4.8) originally assuming that all tethers are similar with equal tension. Then the solution (4.9) is calculated to return the tension in the tether or the total tension of all tethers.

4.4 ODE systems

4.4.1 Full version with power calculation

The ODE solvers of MATLAB require that the ODE system is presented as a function that takes time and a vector with all system variables as parameter together with other optional parameters and returns the time derivatives of all system variables as one vector. In this example simulation this ODE system function is called KiteSystem6P and it uses the dimensionless time τ as the time variable. There are eight system variables that include the dimensionless spherical coordinates ρ, θ and ϕ together with their first derivatives with respect to the dimensionless time τ and also the dimensionless undeformed tether length $\frac{l}{l_0}$ and the dimensionless generated energy $\frac{E_g}{\frac{1}{2}\rho_a A{V_0}^2 l_0}$ for power calculations. In addition to the dimensionless time τ and the system variable vector the control information is given as the additional parameters including the period T_p of the control function, the control coefficients and the number N of odd and even harmonics.

The Value function is called five times to get the mass m of the kite, the characteristic length l_0 of the tether, the average wind speed V_0 at the reference altitude, the density ρ_a of air and the effective wing area A of the kite. Then the time $t = \frac{l_0}{V_0}\tau$ is calculated and FourierControlFun is called two times to get the control angle and the dimensionless tether unwinding speed $\frac{V_L}{V_0}$. The control coefficients are given as a two-row matrix with coefficients for the control angle on the first row and coefficients for the tether unwinding speed on the second row, both defined using the same system. Then the system variables are picked from the vector to give them better names that allow them to be identified easily.

With the values of the two control functions known and good names for the system variables everything is ready for calculating the forces. The Tension function is called

first because the tension is needed twice in force balance and power calculations. Then the force balance is calculated as

$$\mathbf{F^{tot}} = \mathbf{F^{aer}} + \mathbf{F^{line}_{fric}} + \mathbf{F^{gra}} - T\hat{\mathbf{e}}_\mathbf{r}. \tag{4.10}$$

In this $\mathbf{F^{aer}}$ is calculated either using FaerKite or FaerKiteB depending on which control angle definition is chosen. The $\mathbf{F^{line}_{fric}}$ is calculated usually using FaerTether0, but FaerTether1 can be used to make the simulation computationally heavy. FaerTether1 is usually not necessary for accuracy because the component along the tether is small and the aerodynamic coefficients aren't so accurate anyway with dependence on Reynolds number in reality. Then $\mathbf{F^{gra}}$ is calculated using the Fgra function and the tension T was calculated before.

Then in the last part of the function the derivatives of all system variables with respect to the dimensionless time τ are calculated and returned as one vector. The derivatives of ρ, θ and ϕ to be returned are just the other system variables ρ_τ, θ_τ and ϕ_τ, respectively. Then the derivatives of ρ_τ, θ_τ and ϕ_τ are calculated using (3.12) neglecting the inertia of the tethers and the derivative of the dimensionless undeformed tether length

$$\left(\frac{l}{l_0}\right)_\tau = \frac{1}{l_0}\frac{dl}{dt}\frac{dt}{d\tau} = \frac{1}{l_0}V_L\frac{l_0}{V_0} = \frac{V_L}{V_0}$$

is just the dimensionless tether unwinding speed $\frac{V_L}{V_0}$ defined by the control function. Then the derivative of the dimensionless generated energy

$$\left(\frac{E_g}{\frac{1}{2}\rho_a A {V_0}^2 l_0}\right)_\tau = \frac{1}{\frac{1}{2}\rho_a A {V_0}^2 l_0}\frac{dE_g}{dt}\frac{dt}{d\tau} = \frac{1}{\frac{1}{2}\rho_a A {V_0}^2 l_0}P\frac{l_0}{V_0} = \frac{P}{\frac{1}{2}\rho_a A {V_0}^3}$$

is the dimensionless power that is calculated as

$$\frac{P}{\frac{1}{2}\rho_a A {V_0}^3} = \frac{T}{\frac{1}{2}\rho_a A {V_0}^2}\frac{V_L}{V_0}, \tag{4.11}$$

where E_g is the generated energy and P is the instantaneous power of the kite generator. It's worth to notice that the factor $\frac{1}{2}\rho_a {V_0}^2$ used to make the energy and power dimensionless is the dynamic pressure of the reference wind speed V_0. The power calculation is simplified by using the same tether tension as in the force balance calculation neglecting the change of tension along the tether due to the gravity and aerodynamic force acting on the tether.

4.4.2 Simple version with constant tether length

A simpler version of the ODE system is also included in this example code for simulating kite flight with constant tether length quicker. This simpler ODE system function is called KiteSystem4 and it has the same parameters as KiteSystem6P, but with only four system variables in the system variable vector instead of eight in KiteSystem6P and just one row for control angle in the control coefficient matrix. The four system variables are the spherical coordinates θ and ϕ together with their first derivatives with respect to the dimensionless time τ because ρ is always one with constant tether length.

The Value function is called four times to get the mass m of the kite, the constant length l_0 of the tether, the average wind speed V_0 at the reference altitude and the density ρ_a of air. Then the time $t = \frac{l_0}{V_0}\tau$ is calculated and FourierControlFun is called to get the control angle. Then the system variables are picked from the vector to give them better names that allow them to be identified easily, and the constant values $\rho = 1$ and $\rho_\tau = 0$ are defined. Then the force balance at the kite is calculated using (4.10) by calling the same force functions as in KiteSystem6P, but the tether tension is neglected because forces in the radial direction along $\hat{\mathbf{e}}_\mathbf{r}$ aren't needed at all.

Then in the last part of the function the derivatives of all system variables with respect to the dimensionless time τ are calculated and returned as one vector. The derivatives of θ and ϕ to be returned are just the other system variables θ_τ and ϕ_τ, respectively. Then the derivatives of θ_τ and ϕ_τ are calculated using (3.12) that simplifies to the form

$$\begin{aligned} \theta_{\tau\tau} &= {\phi_\tau}^2 \sin\theta\cos\theta + \tfrac{l_0}{m{V_0}^2} F_\theta^{\text{tot}} \quad \text{and} \\ \phi_{\tau\tau} &= -2\theta_\tau\phi_\tau \tfrac{\cos\theta}{\sin\theta} + \tfrac{l_0}{m{V_0}^2} \tfrac{F_\phi^{\text{tot}}}{\sin\theta} \end{aligned} \tag{4.12}$$

in this case with constant tether length. This simple version doesn't model the effects of tether elasticity and shape to the radial coordinate of the kite, but those effects can be simulated by using KiteSystem6P with constant tether length.

4.5 Initial conditions

The initial velocity and the initial dimensionless radial coordinate of the kite are calculated in a function called InitialConditions for simulations using KiteSystem6P. This function gets the initial values of the spherical coordinates θ and ϕ as parameters together with the direction angle η of the initial velocity. The direction angle η is defined so that $\eta = 0$ means launching the kite to the left in the direction of $\hat{\mathbf{e}}_\phi$ and $\eta = \frac{\pi}{2}$ means launching the kite upwards in the direction of $-\hat{\mathbf{e}}_\theta$ when viewed

from the ground station. The tether unwinding speed is assumed to be zero while calculating these initial conditions, so the unwinding of the tether is assumed to start when the simulation starts but not before.

The Value function is called nine times in the beginning to get values on l_0, V_0, A, α_{k0}, number N_l of tethers, d^{line}, $C_\perp$, ρ_a and E. Then the dimensionless radial coordinate ρ is defined to be one temporarily and the WindSpherical function is called to get the dimensionless wind vector in the initial position.

Then the next step is to find the correct angle of attack α_k of the kite, and this requires some iteration. The iteration is done in a while loop where the aerodynamic efficiency G_e of the kite is first calculated using (3.72) with d^{line} multiplied by the number of tethers and then $\frac{\|\mathbf{W_e^p}\|}{V_0}$ is calculated using (3.68) so that (4.5) gives the new angle of attack with the differences due to dimensionless velocities cancelling. The iteration stops when the change in the angle of attack is smaller than 10^{-4} and 10 degrees seems to work as the initial guess.

The direction vector $\hat{\mathbf{t}}_\perp$ for the initial velocity has $-\sin\eta$ as the θ component and $\cos\eta$ as the ϕ component for this definition of the direction angle η. Then the last two components of the dimensionless wind vector are picked as the perpendicular wind vector and the dimensionless initial perpendicular kite speed $\frac{v_\perp}{V_0}$ is calculated using formulas (3.73) and (3.68) with dimensionless velocities. Then the initial values to be returned are

$$\theta_\tau = -\frac{v_\perp}{V_0}\sin\eta \quad \text{and} \quad \phi_\tau = \frac{v_\perp}{V_0}\frac{\cos\eta}{\sin\theta}$$

with the latter divided by $\sin\theta$ because ϕ_τ is multiplied by it in the velocity vector (3.13).

Then the initial dimensionless radial coordinate ρ_0 of the kite has to be solved first from the equation (3.56) before it can be calculated. To do so the equation (3.56) can be represented as

$$j{\rho_0}^3 + k\rho_0 = 1 \quad \text{with} \quad j = \frac{1}{14}\left(\frac{d^{\text{line}}C_\perp}{4AC_L}\right)^2 {l_0}^2 \quad \text{and} \quad k = 1 - \frac{\rho_a A C_L}{2A^{\text{line}}E}\|\mathbf{W_e^p}\|^2 \tag{4.13}$$

using ρ instead of r. This represented equation (4.13) can then be solved analytically using Maple to get

$$\rho_0 = -\frac{1}{6}12^{\frac{1}{3}}\frac{-Y^{\frac{2}{3}} + k12^{\frac{1}{3}}j}{jY^{\frac{1}{3}}} \quad \text{with} \quad Y = \left(9 + \sqrt{3}\sqrt{\frac{4k^3 + 27j}{j}}\right)j^2 \tag{4.14}$$

as the only real solution.

Then j and k in (4.13) are calculated with $\|\mathbf{W_e^p}\|$ from (3.68). Also d^{line} is

multiplied by the number of tethers in j together with $A^{\text{line}} = N_l \frac{\pi}{4} {d^{\text{line}}}^2$ in k to take into account the effects of all tethers instead of just one in (4.13) originally assuming that all tethers are similar with equal tension. Then the solution (4.14) is calculated to return the initial dimensionless radial coordinate ρ_0 of the kite.

Only the initial velocity of the kite is needed for simulations using KiteSystem4, and this is calculated in a function called InitialVelocity. This function calculates the initial velocity of the kite exactly like the InitialConditions function, only the ρ_0 calculations in the end are missing and thus two calls of the Value function to get ρ_a and E aren't needed.

4.6 Parent functions

The functions presented before are not yet enough for a working simulation because a parent function using the ODE solver is needed. This example code has four different parent functions called TrajectoryPlotCont, TrajectoryPlotContP, SaveFlight and SaveFlightP for different situations. The former two simulate the flight of the kite and plot the trajectory while the simulation runs whereas the latter two save the system variables for later analysis without plotting. The ones with P use the KiteSystem6P model with variable tether length and power calculation whereas the ones without P use the simpler KiteSystem4 model for kite trajectory simulations with constant tether length.

All of these parent functions take one vector as parameter with the period T_p of the controls as first element and the direction angle η of the initial velocity as the last element. Then the functions using the simpler KiteSystem4 model have $2N + 1$ control coefficients in the middle as all other elements of the parameter vector when N even and odd harmonics are used to define the controls, and the functions using the KiteSystem6P model have twice that number of control coefficients with ones for the control angle before the ones for the tether unwinding speed. This parameter vector is the only parameter except TrajectoryPlotContP gets the number of orbits to be simulated as another parameter because it counts the number of orbits flown by the kite as an additional feature.

The information returned by the parent functions depends on the function. SaveFlight functions return a data matrix with values of the dimensionless time τ and all system variables on each row so that the rows returned by SaveFlight have $[\tau, \theta, \phi, \theta_\tau, \phi_\tau]$ and the rows returned by SaveFlightP have

$$[\tau, \theta, \phi, \theta_\tau, \phi_\tau, \rho, \rho_\tau, \frac{l}{l_0}, \frac{E_g}{\frac{1}{2}\rho_a A {V_0}^2 l_0}].$$

However, TrajectoryPlotCont just returns the dimensionless flight time before crash and TrajectoryPlotContP returns power data as a matrix with the time used for one

orbit and the average power of that orbit on each row.

All these parent functions use the ode45 solver of MATLAB to integrate the ODE system because according to MATLAB documentation [12] it is a good one for most problems and it seems to work OK here. The solver is called in a while loop until the kite crashes or maximum data matrix size limit is exceeded in SaveFlight functions or the desired number of orbits is exceeded in the TrajectoryPlotContP function. The Value function has two tolerances and two time steps defined for the ODE solver and is called five times in the beginning to get all these and N. The two tolerances are the absolute and relative tolerance for ODE solver and the time steps using the dimensionless time are the initial step for ODE solver and the time span for each solver run in the while loop.

Then the length of the parameter vector is checked before the control function data is picked from the parameter vector to the format used in KiteSystem4 or KiteSystem6P. Then the next step is to define the initial conditions. The Value function is called twice to get the initial values for θ and ϕ, and then the InitialVelocity function is called to get the initial values for θ_τ and ϕ_τ and complete the set for simulations using the simpler KiteSystem4 model. For simulations using the KiteSystem6P model the InitialConditions function is called instead to get the initial value of ρ in addition to the initial values of θ_τ and ϕ_τ, and then the three other missing ones are easy to define because ρ_τ is zero due to the assumption of initially constant tether length, $\frac{l}{l_0}$ is one and the generated energy is zero initially. Then the initial conditions are collected in the initial system variable vector and the time span for the first solver run is defined from zero to the time span parameter from the Value function.

Then the ODE solver options are defined using the odeset function of MATLAB. The values defined in the Value function are used as the absolute and relative tolerance and the initial time step, but the events are trickier to define. The events are defined in a separate function that gets the same parameters as the ODE system function and returns event information like described in MATLAB documentation for odeset function [12]. The example code uses two different event definition functions called StoppingCriteriaP for TrajectoryPlotContP and StoppingCriteria for others. StoppingCriteria detects only kite crashes, so it returns θ subtracted from its safety limit defined in the Value function as the event function, one as the second event property to stop the simulation when the kite crashes and then zero as the third event property to detect all zeros of event function. StoppingCriteriaP detects kite crashes as the second event, but it also detects start of new orbit as first event with ϕ as the event function, one as the second event property to stop the simulation for orbit counting and one as the third event property to detect only zeros of ϕ with increasing ϕ.

Then everything is ready for the solver loop after defining some variables for data storage and loop management. In all parent functions the solver loop is a while loop with the ode45 solver called first inside the loop, crash event detected after plotting or saving the data and then initial conditions and time span defined in the end for next loop. Parameters for the ode45 solver are a handle to the ODE system function, time span, initial condition vector, options and then the extra parameters for the ODE system function. The ode45 solver then returns the values of the dimensionless time τ and system variable vectors for each step and also each detected event together with event indices. The SaveFlight functions save all the values of the dimensionless time τ and system variable vectors in the end of the data matrix and break the loop due to kite crash when some event time is returned, and that's all that happens inside the solver loop of the SaveFlight functions. The initial conditions for the next solver run are just the last system variable vector returned by the solver, and the time span for the next solver run is from the last value of τ returned by the solver to that plus the time span parameter.

The solver loop of the TrajectoryPlotCont function is similar to the SaveFlight functions except that the data is plotted instead of saving. Horizontal angle $-\phi$ is plotted at the x-axis and angle of inclination $\frac{\pi}{2}-\theta$ at the y-axis to get the trajectory look like viewed from the ground station with θ positive downwards and ϕ positive to the left. The TrajectoryPlotContP function has similar solver loop with plotting, but that's inside another while loop to count the orbits of the kite, and also the event detection is different. The solver loop of TrajectoryPlotContP is broken when either of the events in StoppingCriteriaP is detected so that the crash event is identified from the event index and information of crash is saved in a separate variable.

The outer loop of TrajectoryPlotContP includes the power data calculations after running the solver loop first. The time used for the orbit is calculated as $\frac{l_0}{V_0}$ times the difference of the last dimensionless event time and previous saved one. Then the average power is calculated by multiplying the dimensionless generated energy $\frac{E_g}{\frac{1}{2}\rho_a A {V_0}^2 l_0}$ with $\frac{1}{2}\rho_a A {V_0}^2 l_0$ to get the generated energy E_g and then dividing with the time used for the orbit to get the average generated power P for the orbit in watts. Then the time used for the orbit and the average generated power P for the orbit are saved in the power data matrix and the outer loop is broken in case of kite crash. Of course, the Value function is called four more times before the loops to get the values of l_0, V_0, ρ_a and A needed in these power calculations. Then the initial conditions for the next outer loop are the system variable vector for last event with the generated energy changed to zero, and the time span is from the value of τ for last event to that plus the time span parameter. Then in the end of the outer loop the value of τ for last event is saved as the starting time of orbit for the next outer loop, and that's all in the TrajectoryPlotContP function.

5. SIMULATION TESTS

5.1 System properties

The example simulation code presented before can then be tested to hopefully get some interesting results. The two test situations study the stability of orbits of a $10\,\mathrm{m}^2$ kite flown with constant tether length. In the first test different periods of the control function are tested using the full ODE system model, and in the second test the simple ODE model is used to try different amplitudes of control function. All parameter values used in these example simulations are collected in Table 5.1.

Parameter	Symbol	Tests 1–2
Kite wing area	A	$10\,\mathrm{m}^2$
Angle of attack in overhead position	α_{k0}	$-2°$
Kite mass	m	$4\,\mathrm{kg}$
Initial tether length	l_0	$100\,\mathrm{m}$
Number of tethers	N_l	2
Tether diameter	d^{line}	$3\,\mathrm{mm}$
Normal reaction coefficient	$C_\perp$	1.2
Longitudinal reaction coefficient	$C_\parallel$	0.02
Density of tether material	ρ_l	$970\,\mathrm{kg/m}^3$
Tensile modulus of tether material	E	$89\,\mathrm{GPa}$
Wind speed at reference altitude	V_0	$7\,\mathrm{m/s}$
Wind model		power law
Acceleration due to gravity	g	$9.81\,\mathrm{m/s}^2$
Air density	ρ_a	$1.23\,\mathrm{kg/m}^3$
Initial value of θ	θ_0	1.1517
Initial value of ϕ	ϕ_0	0
Maximum value of θ	$\theta_{\max}$	$80°$

Table 5.1: Parameter values used in numerical examples.

All the example simulations use $N = 1$ as the order of control function definition to keep the parameter vectors simple to read. Then the absolute and relative tolerance used in the example simulations are 10^{-8} and the initial time step is 10^{-4} and time span for each solver run is 0.1.

5.2 Test 1: Period of control function and stabilization

The first test uses the full KiteSystem6P model with constant tether length to test different control functions. In this test all the function options and constant values are just like defined in the example code, so the control angle ψ and the simpler tether drag approximation are used. The $10\,\mathrm{m}^2$ kite used is in the size class of a large traction kite, the two tethers are 100 m long and the reference wind speed is not so high at 7 m/s with the wind gradient power law (4.2) used. Thus, the conditions are not so far from normal traction kite flying except the tethers are longer. All the system parameters defined in the Value function are listed in Table 5.1.

Let's use a direction angle of $\eta_1 = \arctan\frac{5.7875}{4.9674} \approx 0.8615$ for the initial velocity and just try to run the TrajectoryPlotContP function with the parameter vector $[2.9, 0, 0, -0.077, 0, 0, 0, \eta_1]$ so that the period T_p is 2.9 seconds and the coefficient b_1 is -0.077 in (4.1) with $N = 1$ and constant tether length. Let's use 1000 as the maximum number of orbits so that the limit isn't reached too soon. This gives the kite trajectory shown in Figure 5.1.

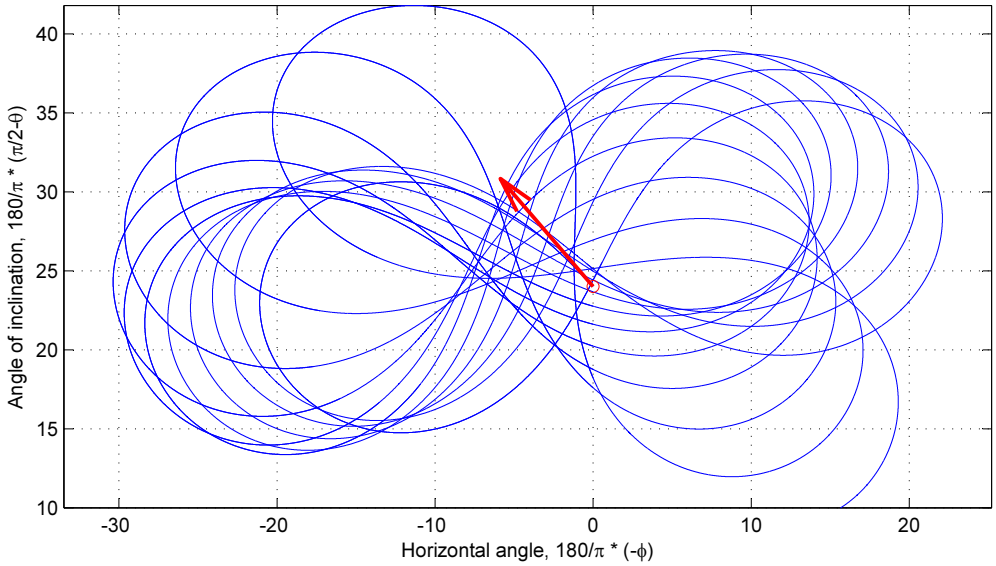

Figure 5.1: Kite trajectory in test 1 with $T_p = 2.9$ s. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

The kite flies almost upwards after the first loop to the left, so looks like it turns a bit too much. That makes the orbit move to the left and then tilt quickly when the center of the orbit is almost 10° off. This tilting of the orbit then causes crash. Thus, let's try a bit shorter period of 2.75 seconds so that the kite doesn't turn so long and so much. Unfortunately, this just makes the kite drop down like seen in Figure 5.2.

Now the orbits look nice, but the kite just drops out of sky after a short flight. Maybe some period between these two first values might keep the kite flying longer

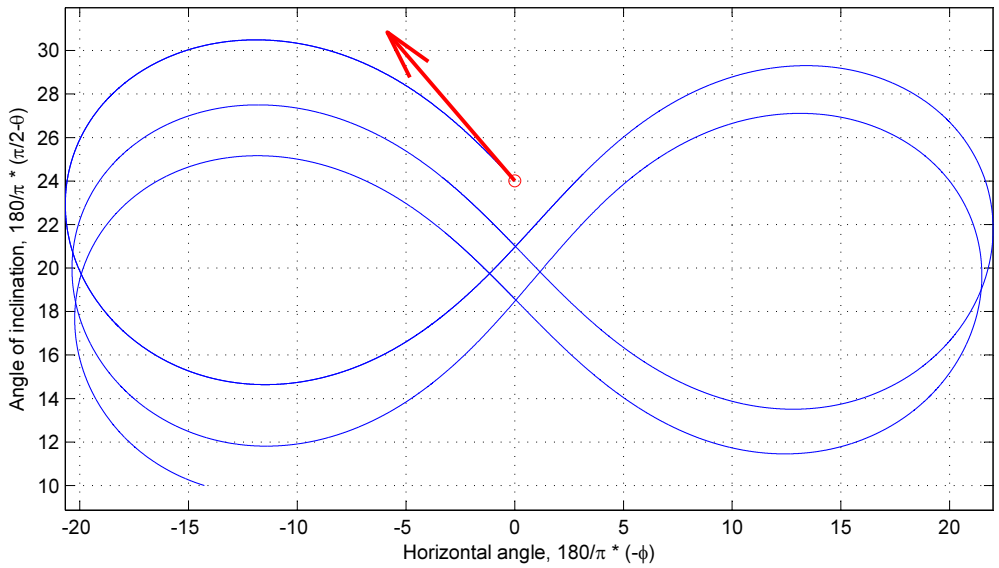

Figure 5.2: Kite trajectory in test 1 with $T_p = 2.75$ s. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

and give more interesting results. Thus, let's go back towards the first period that gave us longer flight and try with $T_p = 2.85$ s to get a trajectory seen in Figure 5.3.

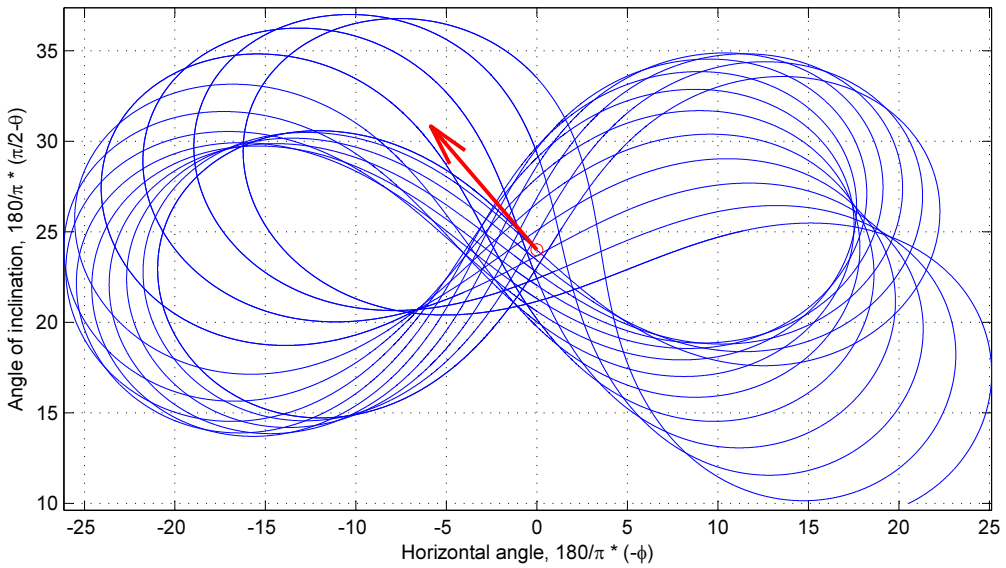

Figure 5.3: Kite trajectory in test 1 with $T_p = 2.85$ s. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

Unfortunately, the kite still crashes after just a bit longer flight than with $T_p = 2.9$ s. However, the orbits are closer to each other, so $T_p = 2.85$ s looks better than $T_p = 2.9$ s. Let's try with $T_p = 2.8$ s then to see what happens with still a bit shorter period. Now something very interesting happens because the kite doesn't crash anymore and seems to stabilize towards some trajectory. Simulating over 1000 orbits to hit the maximum number of orbits takes a lot of time, so stopping the

simulation with Ctrl+C in MATLAB main window is a better option. The trajectory from the beginning to manually stopping the simulation is shown in Figure 5.4.

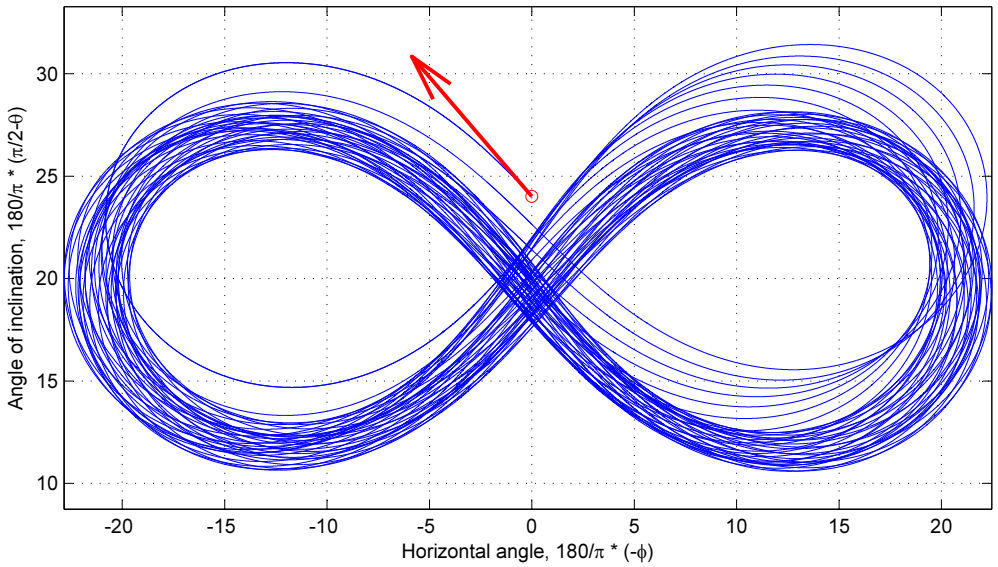

Figure 5.4: Kite trajectory in test 1 with $T_p = 2.8$ s. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

It's very interesting to see whether the kite really finds some stable orbit and stays there. That can be seen by letting the simulation run significantly longer and cleaning the plot window while the simulation runs. Cleaning the plot window after a long simulation actually shows that the kite has found a stable orbit and seems to follow it forever like seen in Figure 5.5 with several orbits on top of each other.

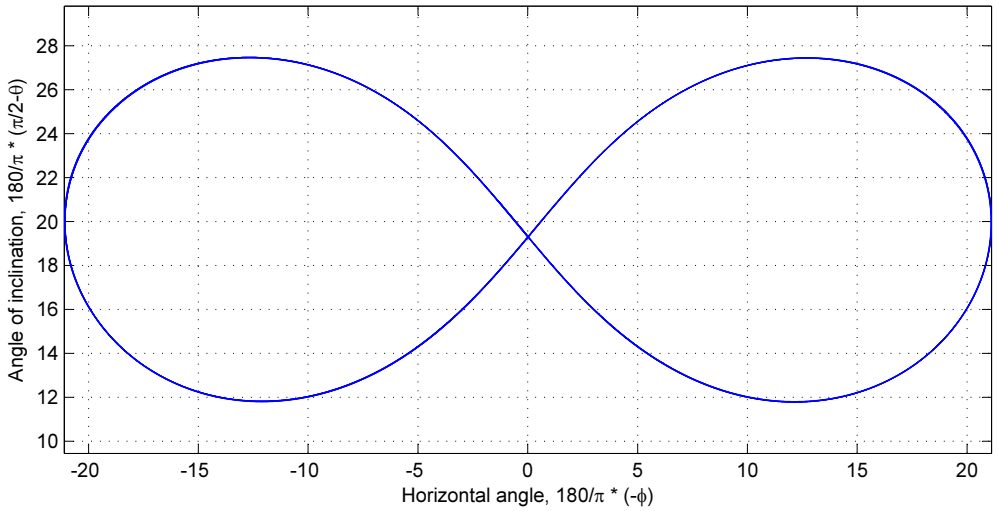

Figure 5.5: Final kite trajectory in test 1 with $T_p = 2.8$ s after a long simulation.

Thus, looks like that some stable orbits exist at least with this model in these conditions. The sinusoidal control input can keep the kite flying forever without any changes in conditions so that the kite follows some kind of natural and stable orbit.

The final stable orbit looks like a lying eight with smooth shape like is expected with smooth sinusoidal control function.

Then it's interesting to see what happens with just a bit shorter period of 2.79 seconds. This is shown in Figure 5.6.

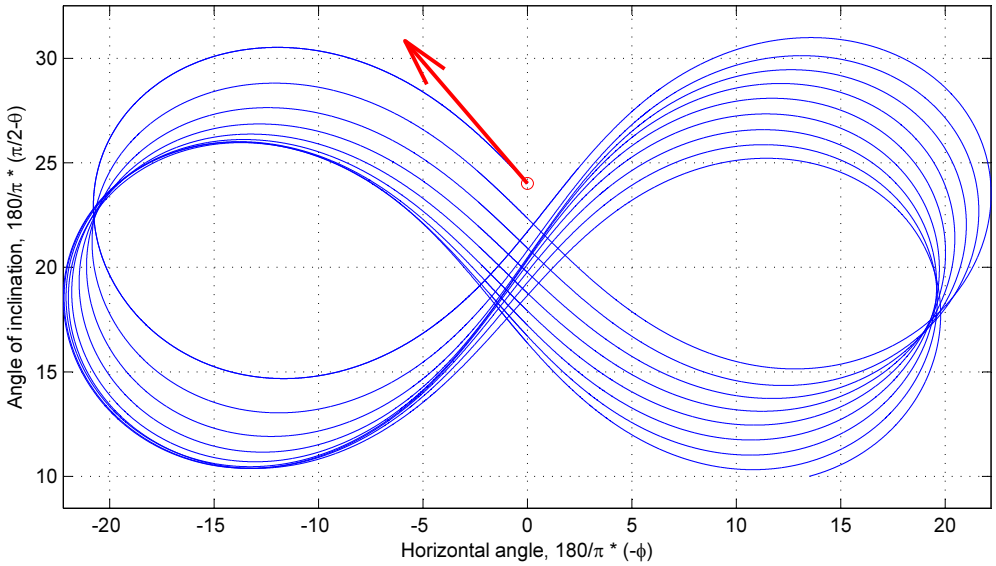

Figure 5.6: Kite trajectory in test 1 with $T_p = 2.79$ s. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

The shape of the orbits is pretty close to the stable one, but still the trajectory drops down and the kite crashes after the orbit starts to tilt slightly to another direction than initially. Thus, even a bit shorter period than the period for the stable orbit seems to make the trajectory drop down causing a crash.

Then it's still interesting to try with a bit longer period than the period for the stable orbit like $T_p = 2.81$ s. The result with $T_p = 2.81$ s is really interesting because in this case the orbit keeps moving left and right while tilting in alternating directions. This oscillation seems to slowly diverge with the kite getting slightly lower after each tilting period, and this likely causes a crash eventually. However, the kite seems to fly a long time before crash, so the simulation was stopped with Ctrl+C to get the trajectory seen in Figure 5.7 before everything gets covered by new orbits.

Similar tilt oscillation happens with $T_p = 2.8$ s, but in that case the oscillation dampens and the kite stabilizes to the final orbit, but $T_p = 2.81$ s makes that tilt oscillation slowly diverge. Thus, the tilt oscillation seems to indicate slight overturning due to a bit too long period. The time span of 0.1 used for each solver run in these tests is about half of one orbit, and the orbit counting gives periods of about 2.8 seconds that matches the periods of the control function, of course.

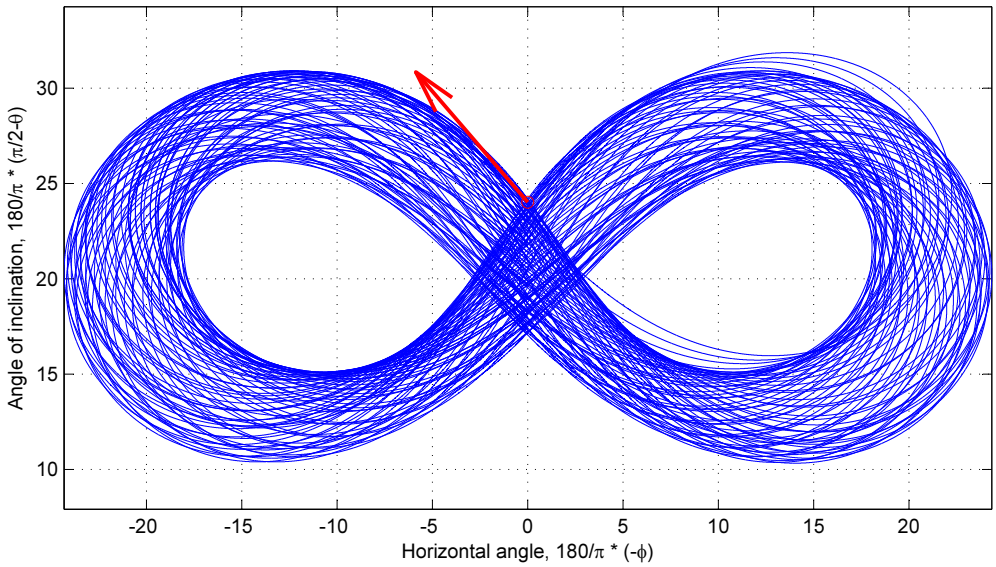

Figure 5.7: Kite trajectory in test 1 with $T_p = 2.81$ s. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

5.3 Test 2: Effects of control amplitude with simple model

The second test uses the same conditions and settings as the first test except that the TrajectoryPlotCont function and thus the simpler KiteSystem4 model is used. Thus, all the function options and constant values are still exactly like in the example code, just a different parent function is used. Let's try to run the TrajectoryPlotCont function with the parameter vector $[2.8, 0, 0, -0.077, \eta_1]$ first with the result is shown in Figure 5.8.

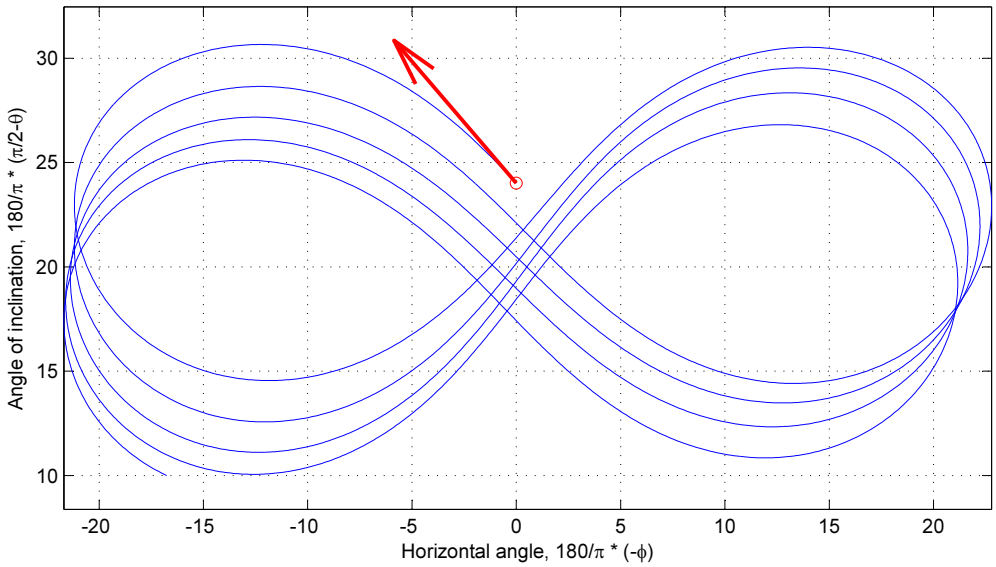

Figure 5.8: Kite trajectory in test 2 with $b_1 = -0.077$. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

This test has the same control function and initial angle as the stable orbit found in the first test to get an idea about the differences of the two models. However, the kite just drops out of sky like with too small period and too little turning. Thus, the two different ODE models give slightly different results likely due to the differences in the tether length model. It's interesting to see whether the orbit stabilizes again after some small correction, and it's more interesting to modify the amplitude of the control function now instead of the period. The amplitude of the control function should be increased to make the kite turn more, so a small correction to $b_1 = -0.0775$ could be good as the next test and the result is shown in Figure 5.9.

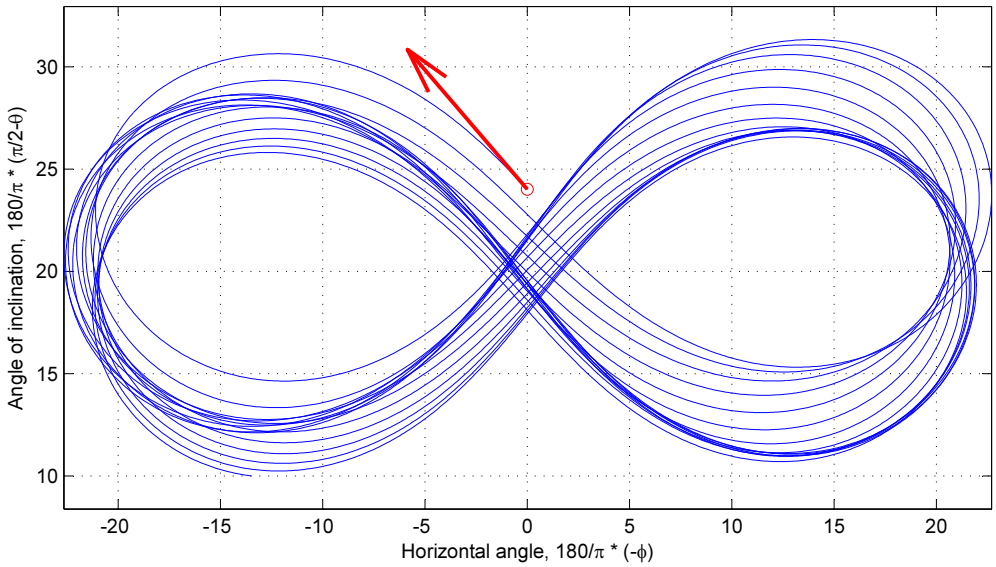

Figure 5.9: Kite trajectory in test 2 with $b_1 = -0.0775$. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

Now the kite completes a lot more orbits but still crashes due to dropping trajectory. This case also has some tilt oscillation visible with many orbits close to each other when the tilt direction changes. Looks like that still a bit tighter turns are needed to keep the kite flying, so $b_1 = -0.078$ could be the next test. This gives an interesting result seen in Figure 5.10.

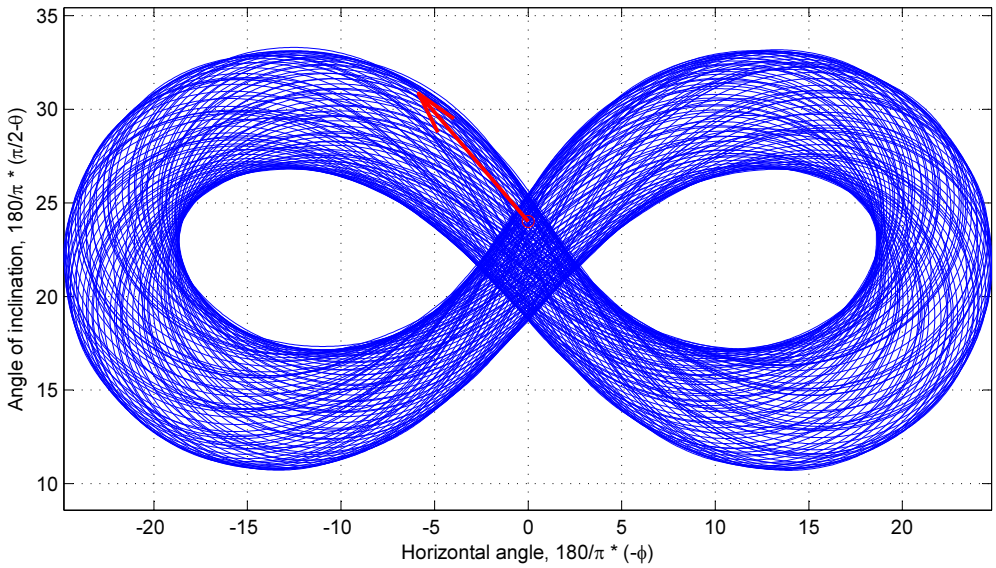

Figure 5.10: Kite trajectory in test 2 with $b_1 = -0.078$. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

Figure 5.10 is taken after stopping the simulation with Ctrl+C before the figure gets too many orbits, but after running the simulation for a long time the situation is still the same like seen in Figure 5.11 that shows some orbits after cleaning the plot window.

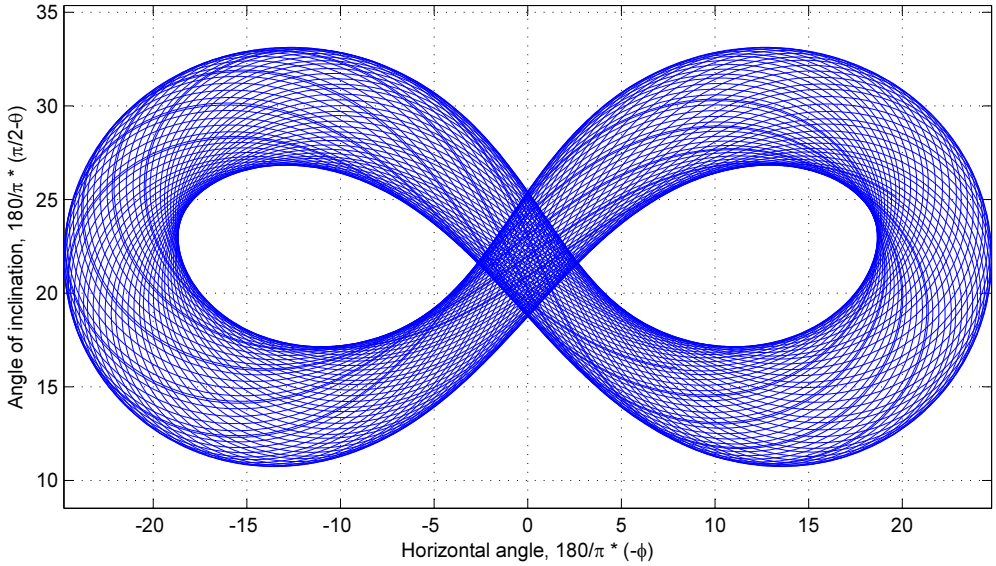

Figure 5.11: Part of the kite trajectory in test 2 with $b_1 = -0.078$ after a long simulation.

The orbit just continues to move around within the limited area so that one part of the orbit always touches the outer edge of that area and the opposite part touches the inner edge of that area in the hole. After moving around once the orbit almost returns to the same position again and then almost follows the previous orbits like seen in the lines close to each other with the first and last orbits almost overlapping.

In this case the orbit moves around quite a lot, but doesn't seem to converge or diverge. Anyway, this kind of oscillation seems to indicate overturning, so let's try with $b_1 = -0.0777$ next. The result is shown in Figure 5.12 with the simulation stopped using Ctrl+C just after the plot doesn't change anymore with all new orbits overlapping older ones.

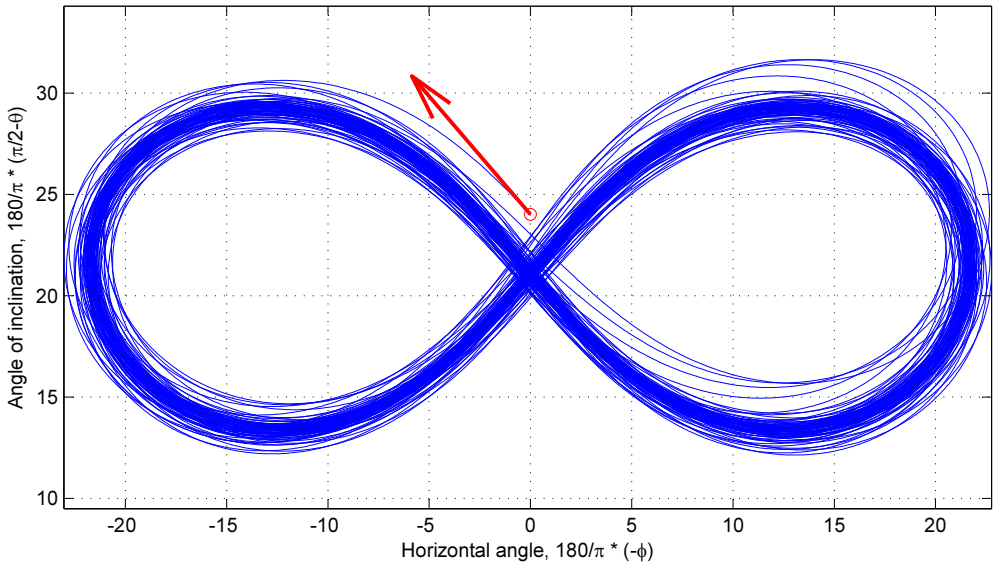

Figure 5.12: Kite trajectory in test 2 with $b_1 = -0.0777$. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

In this test the orbit stabilizes faster than seen in the first test so that the oscillation dampens quite fast. After running the same simulation again for a long time the kite still follows that stable orbit shown in Figure 5.13 with many overlapping orbits after cleaning the plot window.

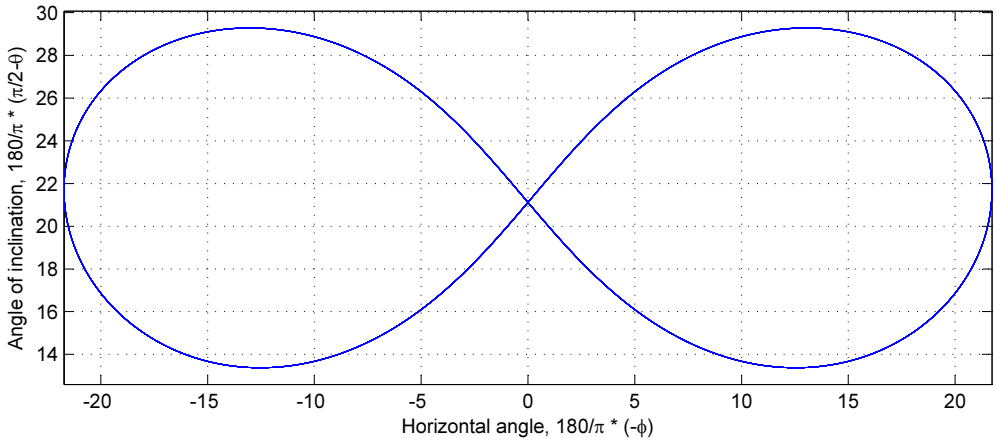

Figure 5.13: Final kite trajectory in test 2 with $b_1 = -0.0777$ after a long simulation.

It's interesting to see that this stable orbit is placed a bit higher than the one in the first test so that there is more safety margin left between it and the horizon.

The kite also was a lot farther from crash before the orbit stabilized than in the first test, and this orbit seems more stable with significantly faster dampening of tilt oscillation. Thus, the simpler ODE model might be a bit more stable or this control function just happens to work better in these conditions than the stable control function found in the first test.

However, it's still interesting to test with $b_1 = -0.0776$ because the results for $b_1 = -0.0775$ and $b_1 = -0.0777$ are so different. The result of this test is shown in Figure 5.14.

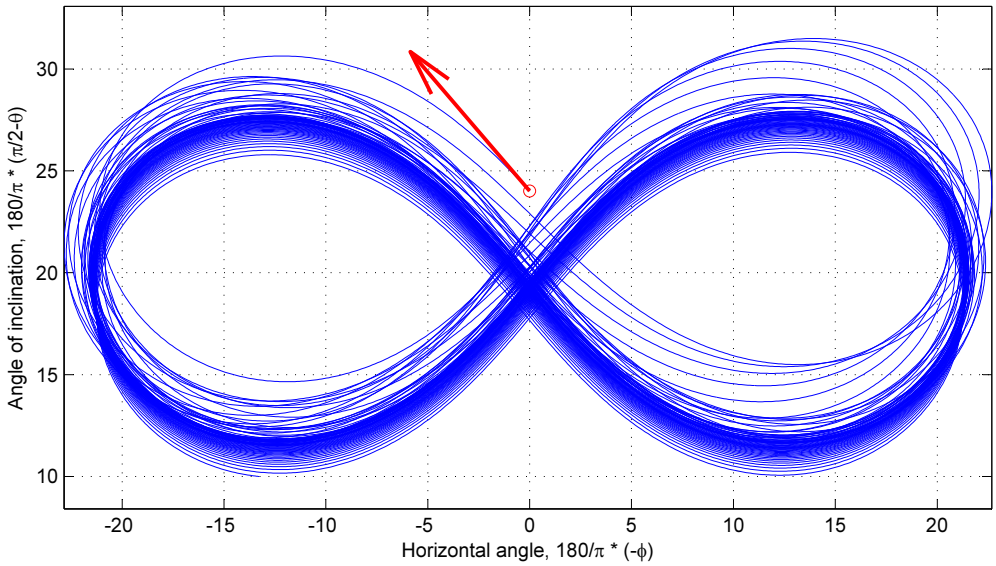

Figure 5.14: Kite trajectory in test 2 with $b_1 = -0.0776$. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

This result is very interesting because the orbit seems to first stabilize quite fast with the tilt oscillation dampening even faster than with $b_1 = -0.0777$. The kite seems to follow that orbit for some time, but then the orbit starts to drop with increasing rate and the kite crashes while still following almost symmetric orbit without any tilt. Then with $b_1 = -0.07761$ the kite follows that stable-looking orbit a lot longer, but that orbit still starts to drop very slowly with increasing rate and the kite crashes after completing many orbits. However, with $b_1 = -0.07762$ the orbit stabilizes significantly faster than with $b_1 = -0.0777$ like seen in Figure 5.15 and the kite still follows the stable orbit of Figure 5.16 exactly in a very long simulation.

It's interesting to see that the stable orbit with $b_1 = -0.07762$ is placed a bit lower than the stable orbit with $b_1 = -0.0777$. This test also shows that the difference between fast stabilization of the orbit and kite crash can be really small. Less turning seems to dampen the tilt oscillation faster, but the stability can be lost suddenly with the orbit dropping with increasing rate. On the other hand, stable orbits

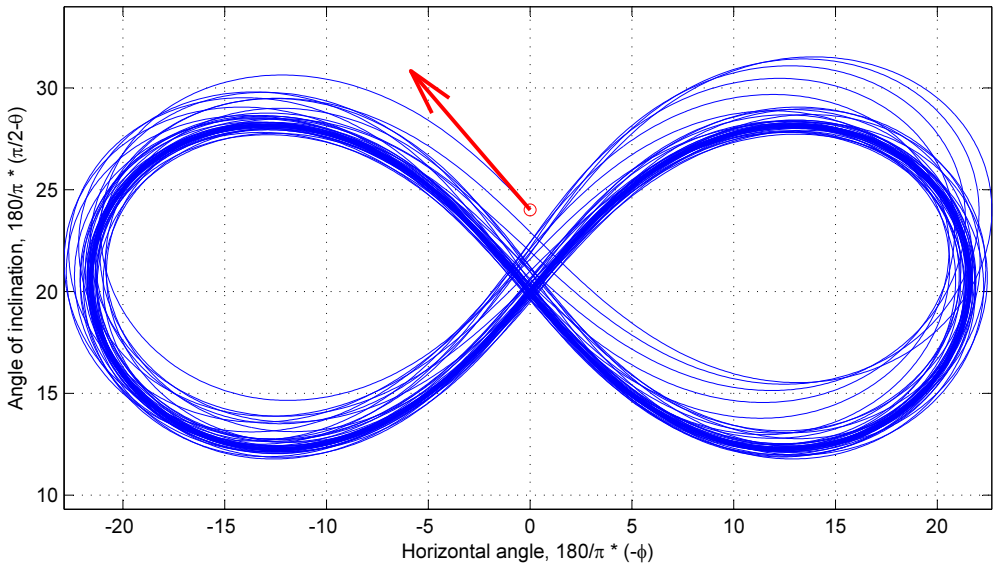

Figure 5.15: Kite trajectory in test 2 with $b_1 = -0.07762$. The blue line is the kite trajectory, and the red circle and arrow mark the initial position and direction of the kite.

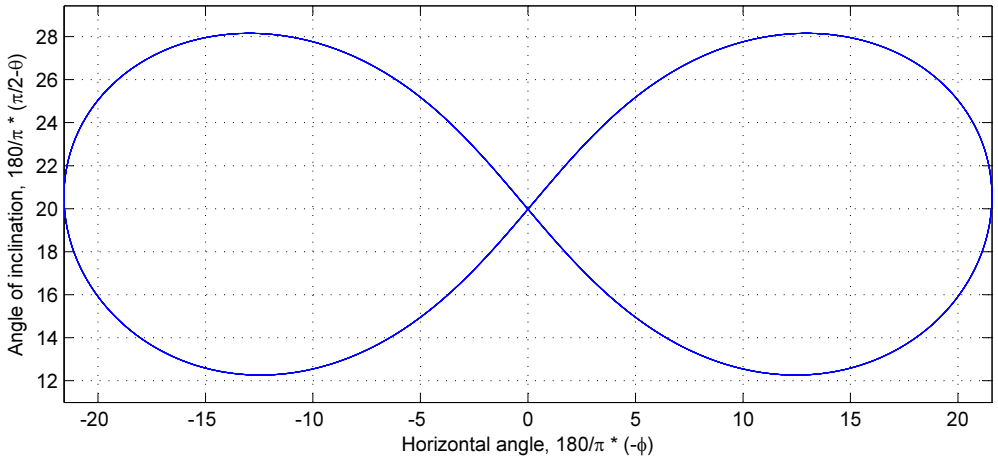

Figure 5.16: Final kite trajectory in test 2 with $b_1 = -0.07762$ after a very long simulation.

can be found also with tighter turns at higher altitude. Tighter turns seem to still allow stability but increase the tilt oscillation so that the oscillation doesn't dampen anymore with enough overturning. Actually, with $b_1 = -0.0781$ the kite just avoids crash in the beginning and the orbit oscillates with even larger amplitude than with $b_1 = -0.078$. However, the orbit still stays within the limited area that almost touches the safety limit of θ, and thus the kite seems to still fly forever without crashing. Then with $b_1 = -0.0782$ the kite crashes after not so many orbits, so that's just too much overturning.

6. CONCLUSION

The kite simulator presented here seems to work OK and is useful for looking at kite trajectories with pre-defined control functions both when flying the kite with constant tether length and when generating energy by letting the kite pull and unwind the tether. The kite trajectories plotted by the simulation look very interesting in some cases and it's interesting to look at them while the kite flies in the simulation. On the other hand, data saving allows further analysis of the flight like plotting the kite speed or angle of attack or trying to find parameters of the final orbit. This simulation should scale well from small prototypes to large kite generators thanks to the dimensionless system variables that should also keep computing times similar for different tether lengths. Changing the system properties including the number of tethers is easy because all of them are collected in one place.

The stability phenomena observed in the first simulation tests with constant tether length and pre-defined sinusoidal control function are very interesting. Looks like that at least in some conditions the kite has some kind of natural stable orbits so that the kite stabilizes on them even from different initial conditions and stays there until the conditions change. Then very small differences in the control function can cause crash instead of fast stabilization if the kite turns too little, and turning the kite slightly too much made the orbit move around interestingly within a limited area. These stability phenomena still have many interesting questions to research, for example adjusting control period and amplitude simultaneously to try to get stable orbits with different sizes and testing different disturbances to see how well the kite can stabilize back on the natural stable orbit without any feedback. Unfortunately, tether length changes while generating energy prevent such stability, but it's still possible to get many lying-eight orbits with shape similar to the stable orbits in the energy generation phase and start the retraction phase before the kite gets out of control when using pre-defined sinusoidal control function, but some feedback is likely necessary to control the retraction phase. Thus, such lying-eight orbits might be useful in kite generators to control the kite easier with the help of natural stability.

The simulation model presented here is basically quite simple as a point mass model without any modeling of kite rotations. The kite is assumed to always align with the airflow around it so that sideslip isn't modeled and this model and simu-

lation can't be used for kites controlled by systems causing sideslip like drag flaps. The bank angle of the kite is defined directly by the control input without any inertia or control force modeling, so this model works well only for kites controlled by bank angle and with good directional stability. The angle of attack of the kite is also included in the simulation, but without any kite rotation and longitudinal balance models the kite angle of attack can be modeled properly only in two extreme cases. One case used here is the assumption that the wing always stays in the same angle with respect to the position vector, and this situation is closer to reality when the bridle point around which the kite rotates is far from the wing compared to the chord of the wing and the tether is tight and thus close to straight. The other case is when the angle of attack is kept constant with some active control system in the kite because this can be simulated with constant lift and drag coefficients in the kite aerodynamics model. Constant kite angle of attack is actually a great option in large complex kite generators to keep the optimal angle of attack for best possible glide ratio all the time and thus get maximal generated power.

Unfortunately, adding rotation models to the simulation would require more equations of motion and make the model more difficult to understand. There would also be a lot more kite specific properties including aerodynamic stability, control response and movement of aerodynamic center position that would require a lot of work when defining the proper values and functions for each kite including wind tunnel experiments and structural vibration analysis in difficult cases before good accuracy is possible. Then proper rotation modeling also requires detailed modeling of the tether shape to know how the non-straight tether affects the angle of attack of the kite. Then inertia effects of heavier tethers may significantly affect the bank angle of the kite and make the kite turn otherwise than expected, and modeling this effect requires tether dynamics model with possible vibrations. That's why the inertia of the tether is neglected in the presented model because simple effective mass approach won't work well without the tether following the kite as a rigid body. Thus, a simple point mass model like presented here is useful because it's easy to use with just some simple system parameters to define and easy to understand without any long equations that can't be documented well.

The theory section has some results and formulas that may be useful in other models and calculations also. The tether modeling gives an useful simple approximation for the tether drag (3.42) that lets us know how much the tether slows down the kite speed, and then the approximate solution for the tether shape (3.55) lets us estimate the angle of the tether at the kite ($x = r$) from the straight line to know how the tether shape affects the kite angle of attack. Then the tether length equation (3.56) lets us know the connection between the undeformed tether length and the real distance between the two ends of the tether. Then the two control

angle definitions, their relations to the common basis vectors and the aerodynamic force formulas can be useful when using and comparing the two options.

The modular structure of the simulation code allows easy implementation of more detailed models for wind conditions or kite aerodynamics, for example. Testing different wind conditions and kite aerodynamics models would also be useful because the presented models are just simple placeholders that should work well enough in normal situations but won't work well in some cases like higher altitudes and stall situations. Then one possibility for further research is to add an active controller to the system and try to develop controllers that can keep the kite flying well in various conditions with various disturbances. This mathematical model of kite generator could also be useful for model predictive control, but some optimizations and simplifications may be needed for real-time model predictive control applications with limited compute performance.

REFERENCES

[1] I Argatov, P Rautakorpi, and R Silvennoinen. Estimation of the mechanical energy output of the kite wind generator. *Renewable Energy*, 34(6):1525–1532, 2009.

[2] I Argatov and R Silvennoinen. Asymptotic modeling of unconstrained control of a tethered power kite moving along a given closed-loop spherical trajectory. *Journal of Engineering Mathematics*, 72(1):187–203, 2012.

[3] M Canale, L Fagiano, M Ippolito, and M Milanese. Control of tethered airfoils for a new class of wind energy generator. In *45th IEEE Conference on Decision and Control*, pages 4020–4026, 2006.

[4] M Canale, L Fagiano, and M Milanese. Power kites for wind energy generation: Fast predictive control of tethered airfoils. *IEEE Control Systems Magazine*, 27(6):25–38, 2007.

[5] Moritz Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Department of Applied Mathematics, Universität Heidelberg, Heidelberg, 2001. `http://homes.esat.kuleuven.be/~mdiehl/DISS/diehl-diss.pdf`.

[6] Lorenzo Fagiano. *Control of Tethered Airfoils for High-Altitude Wind Energy Generation*. PhD thesis, Politecnico di Torino, Torino, 2009. `http://www.aweconsortium.org/public/downloads/resources/fagiano.pdf`.

[7] Erich Hau. *Wind Turbines: Fundamentals, Technologies, Application, Economics*. Springer, Berlin, 2nd edition, 2006.

[8] S E Hobbs. *A Quantitative Study of Kite Performance in Natural Wind with Application to Kite Anemometry*. PhD thesis, Ecological Physics Research Group, Cranfield Institute of Technology, Cranfield, 1986. `http://dspace.lib.cranfield.ac.uk/handle/1826/918`.

[9] S F Hoerner. *Fluid-Dynamic Drag*. Hoerner Fluid Dynamics, New Jersey, 1965.

[10] M Kaltschmitt, W Streicher, and A Wiese. *Renewable Energy: Technology, Economics and Environment*. Springer, Berlin, 2007.

[11] Miles L Loyd. Crosswind kite power. *Journal of Energy*, 4(3):106–111, 1980.

[12] MathWorks. Documentation center. `http://www.mathworks.se/help/matlab/`, 2013. Viewed November 10, 2013.

[13] P Rautakorpi, I Argatov, and R Silvennoinen. Estimation of the mechanical power of a kite wind generator. In S P Lohani, editor, *Renewable Energy for Sustainable Future*. iConcept Press Ltd., USA, 2013.

[14] David K Schmidt. *Modern Flight Dynamics*. McGraw-Hill, New York, 2012.

[15] P Williams, B Lansdorp, and W Ockels. Optimal crosswind towing and power generation with tethered kites. *Journal of Guidance, Control, and Dynamics*, 31(1):81–93, 2008.

[16] Paul Williams. Optimal wind power extraction with a tethered kite. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.

APPENDIX 1: MATLAB FUNCTIONS

Parent functions

```
function powerdata=TrajectoryPlotContP(params,loops)
% Simulates kite flight until crash, returns average power for each loop
% and plots trajectory of kite
% params are optimization parameters
% params(1) is period of periodic control function
% params(end) is direction angle for initial velocity
% other parameters are coefficients for control function
% loops is number of orbits to simulate

tolRel=Value('tolRel');     %relative tolerance for ODE solver
tolAbs=Value('tolAbs');     %absolute tolerance for ODE solver
step=Value('step');         %initial step for ODE solver
STEP=Value('STEP');         %timespan for each solver run
N=Value('N');               %order of control function definition
coeffNumber=2*N+1;  %number of Fourier coefficients for control function

l0=Value('l0');        %characteristic length of tether [m]
V0=Value('V0');        %average windspeed at reference height [m/s]
airD=Value('airD');    %density of air [kg/m^3]
Akite=Value('Akite'); %effective wing area of kite [m^2]

if length(params) ~= 2*coeffNumber+2
    error('Incorrect number of optimization parameters')
end

period=params(1);           %period of periodic control function
%Fourier coefficients for controls
controlCoeff=[params(2:coeffNumber+1)                  %control angle
              params(coeffNumber+2:2*coeffNumber+1)]; %tether unwinding

theta0=Value('theta0');     %getting initial conditions
phi0=Value('phi0');
[vel0,rho0]=InitialConditions(theta0,phi0,params(end));
theta10=vel0(1);
phi10=vel0(2);
rho10=0;
length0=1;
energy0=0;
%initial conditions
ics=[theta0 phi0 theta10 phi10 rho0 rho10 length0 energy0];
tspan=[0 STEP];             %first timespan for solver
```

```
options = odeset('RelTol',tolRel,'AbsTol',tolAbs,'InitialStep',step,...
                 'Events',@StoppingCriteriaP);  %setting solver options

loopCounter=0;  %for counting orbits
powerdata=[];   %for saving power data
crash=false;    %to get crash information out of the inner loop
prevTE=0;       %for saving dimensionless starting time of orbit

while loopCounter < loops %loop until crash or enough orbits
    while true %solver loop until crash or complete orbit
        [T,Y,TE,YE,IE]=ode45(@KiteSystem6P,tspan,ics,options,period,...
                             controlCoeff,N); %calling ODE solver
        plot(-Y(:,2)*180/pi,(pi/2-Y(:,1))*180/pi); %plotting trajectory
        drawnow
        hold on
        if ~isempty(TE) %detecting crash or next orbit
            if IE(end) == 2 %detecting crash
                crash=true;
            end
            break
        end
        ics=Y(end,:); %initial conditions for next inner loop
        tspan=[T(end) T(end)+STEP]; %timespan for next inner loop
    end

    realPeriod=l0/V0*(TE(end)-prevTE); %calculating time used for orbit [s]
    power=0.5*airD*Akite*V0^2*l0*YE(end,8)/realPeriod; %average power
    powerdata=[powerdata; realPeriod power]; %saving time and power

    if crash %stopping in case of crash
        disp('Kite crashed');
        break
    end

    ics=[YE(end,1:7), 0]; %initial conditions for next outer loop
    tspan=[TE(end) TE(end)+STEP]; %timespan for next outer loop

    loopCounter=loopCounter+1;
    prevTE=TE(end); %updating dimensionless starting time of orbit
end

hold off
```

```
function data=SaveFlightP(params)
% Simulates kite flight until crash and returns flight data
% params are optimization parameters
% params(1) is period of periodic control function
% params(end) is direction angle for initial velocity
% other parameters are coefficients for control function
% each row of data matrix is
% [tau theta phi dtheta/dtau dphi/dtau rho drho/dtau L P],
% tau is dimensionless time

tolRel=Value('tolRel');     %relative tolerance for ODE solver
tolAbs=Value('tolAbs');     %absolute tolerance for ODE solver
step=Value('step');         %initial step for ODE solver
STEP=Value('STEP');         %timespan for each solver run
N=Value('N');               %order of control function definition
coeffNumber=2*N+1;  %number of Fourier coefficients for control function

if length(params) ~= 2*coeffNumber+2
    error('Incorrect number of optimization parameters')
end

period=params(1);           %period of periodic control function
%Fourier coefficients for controls
controlCoeff=[params(2:coeffNumber+1)                  %control angle
              params(coeffNumber+2:2*coeffNumber+1)]; %tether unwinding

theta0=Value('theta0');     %getting initial conditions
phi0=Value('phi0');
[vel0,rho0]=InitialConditions(theta0,phi0,params(end));
theta10=vel0(1);
phi10=vel0(2);
rho10=0;
length0=1;
energy0=0;
%initial conditions
ics=[theta0 phi0 theta10 phi10 rho0 rho10 length0 energy0];
tspan=[0 STEP];             %first timespan for solver

options = odeset('RelTol',tolRel,'AbsTol',tolAbs,'InitialStep',step,...
                 'Events',@StoppingCriteria);  %setting solver options
data=[]; %for saving system data

while size(data,1) < 50000 %solver loop until crash or data size limit
    [T,Y,TE,YE,IE]=ode45(@KiteSystem6P,tspan,ics,options,period,...
                         controlCoeff,N); %calling ODE solver
    data=[data; [T Y]]; %saving system data
    if ~isempty(TE) %detecting crash
        break
```

```
    end
    ics=Y(end,:); %initial conditions for next loop
    tspan=[T(end) T(end)+STEP]; %timespan for next loop
end


function [value,isterminal,direction]=StoppingCriteria(tau,y,period,...
                          controlCoeff,N)
% Event function to detect end of flight

thetaMAX=Value('thetaMAX');  %safety limit for theta
value=thetaMAX-y(1);  %negative when kite flies too close to horizon
isterminal=1;         %simulation ends when value gets negative
direction=0;          %all zeros of value are detected
end


function [value,isterminal,direction]=StoppingCriteriaP(tau,y,period,...
                          controlCoeff,N)
% Event function to detect end of flight and crossing phi=0 to right

value(1)=y(2);           %value of phi to detect start of next orbit
isterminal(1)=1;         %simulation ends when value gets negative
direction(1)=1;          %only zeros with increasing phi are detected

thetaMAX=Value('thetaMAX');  %safety limit for theta
value(2)=thetaMAX-y(1);  %negative when kite flies too close to horizon
isterminal(2)=1;         %simulation ends when value gets negative
direction(2)=0;          %all zeros of value are detected
end
```

```
function time=TrajectoryPlotCont(params)
% Simulates kite flight until crash and plots trajectory of kite
% params are optimization parameters
% params(1) is period of periodic control function
% params(end) is direction angle for initial velocity
% other parameters are coefficients for control function

tolRel=Value('tolRel');     %relative tolerance for ODE solver
tolAbs=Value('tolAbs');     %absolute tolerance for ODE solver
step=Value('step');         %initial step for ODE solver
STEP=Value('STEP');         %timespan for each solver run
N=Value('N');               %order of control function definition
coeffNumber=2*N+1;  %number of Fourier coefficients for control function

if length(params) ~= coeffNumber+2
    error('Incorrect number of optimization parameters')
end

period=params(1);           %period of periodic control function
controlCoeff=params(2:coeffNumber+1);   %Fourier coefficients for controls

theta0=Value('theta0');     %getting initial conditions
phi0=Value('phi0');
vel0=InitialVelocity(theta0,phi0,params(end));
theta10=vel0(1);
phi10=vel0(2);
ics=[theta0 phi0 theta10 phi10];  %initial conditions
tspan=[0 STEP];             %first timespan for solver

options = odeset('RelTol',tolRel,'AbsTol',tolAbs,'InitialStep',step,...
                 'Events',@StoppingCriteria);  %setting solver options

while true %solver loop until crash
    [T,Y,TE,YE,IE]=ode45(@KiteSystem4,tspan,ics,options,period,...
                         controlCoeff,N); %calling ODE solver
    plot(-Y(:,2)*180/pi,(pi/2-Y(:,1))*180/pi); %plotting solved trajectory
    drawnow
    hold on
    if ~isempty(TE) %detecting crash
        break
    end
    ics=Y(end,:); %initial conditions for next loop
    tspan=[T(end) T(end)+STEP]; %timespan for next loop
end
hold off
time=TE; %returning dimensionless flight time
```

```
function data=SaveFlight(params)
% Simulates kite flight until crash and returns flight data
% params are optimization parameters
% params(1) is period of periodic control function
% params(end) is direction angle for initial velocity
% other parameters are coefficients for control function
% each row of data matrix is [tau theta phi dtheta/dtau dphi/dtau],
% tau is dimensionless time

tolRel=Value('tolRel');     %relative tolerance for ODE solver
tolAbs=Value('tolAbs');     %absolute tolerance for ODE solver
step=Value('step');         %initial step for ODE solver
STEP=Value('STEP');         %timespan for each solver run
N=Value('N');               %order of control function definition
coeffNumber=2*N+1;  %number of Fourier coefficients for control function

if length(params) ~= coeffNumber+2
    error('Incorrect number of optimization parameters')
end

period=params(1);           %period of periodic control function
controlCoeff=params(2:coeffNumber+1);   %Fourier coefficients for controls

theta0=Value('theta0');     %getting initial conditions
phi0=Value('phi0');
vel0=InitialVelocity(theta0,phi0,params(end));
theta10=vel0(1);
phi10=vel0(2);
ics=[theta0 phi0 theta10 phi10];  %initial conditions
tspan=[0 STEP];             %first timespan for solver

options = odeset('RelTol',tolRel,'AbsTol',tolAbs,'InitialStep',step,...
                 'Events',@StoppingCriteria);  %setting solver options
data=[]; %for saving system data

while size(data,1) < 50000 %solver loop until crash or data size limit
    [T,Y,TE,YE,IE]=ode45(@KiteSystem4,tspan,ics,options,period,...
                         controlCoeff,N); %calling ODE solver
    data=[data; [T Y]]; %saving system data
    if ~isempty(TE) %detecting crash
        break
    end
    ics=Y(end,:); %initial conditions for next loop
    tspan=[T(end) T(end)+STEP]; %timespan for next loop
end
```

Initial conditions

```
function [vel0, rho0]=InitialConditions(theta0,phi0,eta)
% Calculates initial velocity and rho of kite based on crosswind motion law
% (constant tether length)
% theta0  initial value of theta coordinate
% phi0    initial value of phi coordinate
% eta     direction angle of initial velocity in radians

l0=Value('l0');             %characteristic length of tether [m]
V0=Value('V0');             %average windspeed at reference height [m/s]
Akite=Value('Akite');       %effective wing area of kite [m^2]
AoA0=Value('AoA0'); %angle of attack of the kite in overhead position [deg]
tethers=Value('tethers');   %number of tethers
dTether=Value('dTether');   %diameter of one tether [m]
Cnorm=Value('Cnorm');       %normal reaction coefficient
airD=Value('airD');         %density of air [kg/m^3]
E=Value('E');               %elastic modulus of tether material [Pa]

rho=1;
wind=WindSpherical(V0,l0,rho,theta0,phi0,0);  %wind vector

AoA=10;                %initial guess for angle of attack [deg]
deltaAoA=1;

while abs(deltaAoA) > 10e-4       %iterating to find correct AoA
    Ge=Cl(AoA)/(Cd(AoA)+Cnorm*tethers*dTether*rho*l0/(4*Akite));
    nWep=Ge*wind(1);
    deltaAoA= AoA0 + asin(wind(1)/sqrt(wind(1)^2+nWep^2))*180/pi - AoA;
    AoA=AoA+deltaAoA;
end

%calculating initial velocity
t0=[-sin(eta); cos(eta)];  %direction of initial velocity
windPer=[wind(2); wind(3)];
Vper=t0'*windPer + sqrt(Ge^2*wind(1)^2+(t0'*windPer)^2-norm(windPer)^2);
%returning initial derivatives of theta and phi wrt dimensionless time
vel0=[-Vper*sin(eta) Vper*cos(eta)/sin(theta0)];

%calculating initial radial coordinate
j=1/14*(tethers*dTether*Cnorm/(4*Akite*Cl(AoA)))^2*l0^2;
k=1-airD*Akite*Cl(AoA)*Ge^2*V0^2*wind(1)^2/(2*E*tethers*pi/4*dTether^2);

Y=(9+sqrt(3)*sqrt((4*k^3+27*j)/j))*j^2;
rho0=-(1/6)*12^(1/3)*(-Y^(2/3)+k*12^(1/3)*j)/(j*Y^(1/3));
end
```

```
function vel0=InitialVelocity(theta0,phi0,eta)
% Calculates initial velocity of kite based on crosswind motion law
% (constant tether length)
% theta0  initial value of theta coordinate
% phi0    initial value of phi coordinate
% eta     direction angle of initial velocity

l0=Value('l0');            %characteristic length of tether [m]
V0=Value('V0');            %average windspeed at reference height [m/s]
Akite=Value('Akite');      %effective wing area of kite [m^2]
AoA0=Value('AoA0'); %angle of attack of the kite in overhead position [deg]
tethers=Value('tethers');  %number of tethers
dTether=Value('dTether');  %diameter of one tether [m]
Cnorm=Value('Cnorm');      %normal reaction coefficient

rho=1;
wind=WindSpherical(V0,l0,rho,theta0,phi0,0);  %wind vector

AoA=10;                %initial guess for angle of attack [deg]
deltaAoA=1;

while abs(deltaAoA) > 10e-4       %iterate to find correct AoA
    Ge=Cl(AoA)/(Cd(AoA)+2*Cnorm*rho*l0*dTether/(4*Akite));
    nWep=Ge*wind(1);
    deltaAoA= AoA0 + asin(wind(1)/sqrt(wind(1)^2+nWep^2))*180/pi - AoA;
    AoA=AoA+deltaAoA;
end

%calculating initial velocity
t0=[-sin(eta); cos(eta)];  %direction of initial velocity
windPer=[wind(2); wind(3)];
Vper=t0'*windPer + sqrt(Ge^2*wind(1)^2+(t0'*windPer)^2-norm(windPer)^2);
%returning initial derivatives of theta and phi wrt dimensionless time
vel0=[-Vper*sin(eta) Vper*cos(eta)/sin(theta0)];
end
```

ODE systems

```
function dy=KiteSystem6P(tau,y,period,controlCoeff,N)
% ODE system describing a kite with variable length tethers and one control
% angle
% tau       dimensionless time
% y         system variables [theta phi theta1 phi1 rho rho1 length energy]
% period    period of periodic control
% controlCoeff  control function coefficients as 2-row matrix
% N         order of control function definition

m=Value('m');          %mass of kite [kg]
l0=Value('l0');        %characteristic length of tether [m]
V0=Value('V0');        %average windspeed at reference height [m/s]
airD=Value('airD');    %density of air [kg/m^3]
Akite=Value('Akite'); %effective wing area of kite [m^2]

t=l0/V0*tau;           %time [s]
psi=FourierControlFun(t,period,controlCoeff(1,:),N); %control angle
VL=FourierControlFun(t,period,controlCoeff(2,:),N); %tether unwinding speed

%naming system variables better
theta=y(1);
phi=y(2);
theta1=y(3);
phi1=y(4);
rho=y(5);
rho1=y(6);
length=y(7);
energy=y(8);

%getting tether tension
tension=Tension(length,rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD);

%calculating force balance at kite
force=FaerKite(psi,rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)...
      + FaerTether0(rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)...
      + Fgra(rho,theta,l0,m) - tension*[1; 0; 0];

%returning derivatives of system variables with respect to
%dimensionless time
dy(1)=theta1;
dy(2)=phi1;
dy(3)=phi1^2*sin(theta)*cos(theta) - 2/rho*rho1*theta1...
      + l0/(V0^2*m*rho)*force(2);
dy(4)=-2/rho*rho1*phi1 - 2*theta1*phi1/tan(theta)...
      + l0/(V0^2*m*rho*sin(theta))*force(3);
```

```
dy(5)=rho1;
dy(6)=rho*theta1^2 + rho*phi1^2*sin(theta)^2 + l0/(V0^2*m)*force(1);
dy(7)=VL;
dy(8)=tension*VL/(0.5*airD*Akite*V0^2);
dy=dy';
end


function dy=KiteSystem4(tau,y,period,controlCoeff,N)
% ODE system describing a kite with constant length tethers and one control
% angle
% tau         dimensionless time
% y           system variables [theta phi theta1 phi1]
% period      period of periodic control
% controlCoeff   control function coefficients
% N           order of control function definition

m=Value('m');         %mass of kite [kg]
l0=Value('l0');       %characteristic length of tether [m]
V0=Value('V0');       %average windspeed at reference height [m/s]
airD=Value('airD');   %density of air [kg/m^3]

t=l0/V0*tau;          %time [s]
psi=FourierControlFun(t,period,controlCoeff,N); %control angle

%naming system variables better and adding missing values
rho=1;
theta=y(1);
phi=y(2);
rho1=0;
theta1=y(3);
phi1=y(4);

%calculating forces at kite
Force=FaerKite(psi,rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)...
      + FaerTether0(rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)...
      + Fgra(rho,theta,l0,m);

%returning derivatives of system variables with respect to
%dimensionless time
dy(1)=theta1;
dy(2)=phi1;
dy(3)=phi1^2*sin(theta)*cos(theta) + l0/(V0^2*m)*Force(2);
dy(4)=-2*theta1*phi1/tan(theta) + l0/(V0^2*m*sin(theta))*Force(3);
dy=dy';
end
```

Forces

```
function F=FaerKite(psi,rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)
% Calculates aerodynamic force of kite in spherical coordinate basis
% psi     control angle
% rho,theta,phi     spherical coordinates of kite, rho normalized
% rho1,theta1,phi1  derivatives of spherical coordinates of kite
%                   with respect to dimensionless time
% V0      average windspeed at reference height [m/s]
% l0      characteristic length of tether [m]
% t       time [s]
% airD    density of air [kg/m^3]

Akite=Value('Akite'); %effective wing area of kite [m^2]
AoA0=Value('AoA0'); %angle of attack of the kite in overhead position [deg]

kiteVelocity=[rho1            %dimensionless kite velocity
              rho*theta1
              rho*phi1*sin(theta)];

We=WindSpherical(V0,l0,rho,theta,phi,t)-kiteVelocity; %effective wind
e_w=We/norm(We);  %longitudinal axis of kite

Wep=[0; We(2); We(3)];
e_wp=Wep/norm(Wep);

e_r=[1; 0; 0];
e_sp=cross(e_r,e_wp);
e_s=sin(psi)*e_r - We(1)/norm(Wep)*sin(psi)*e_wp...
    + sqrt(cos(psi)^2-norm(We)^2/norm(Wep)^2*sin(psi)^2)*e_sp;

AoA= AoA0 + asin(We(1)/norm(We))*180/pi;  %angle of attack of kite [deg]

lift=0.5*airD*Akite*Cl(AoA)*V0^2*norm(We)^2;  %magnitude of lift component
drag=0.5*airD*Akite*Cd(AoA)*V0^2*norm(We)^2;  %magnitude of drag component

F=lift*cross(e_w,e_s) + drag*e_w;  %returning the aerodynamic force vector
end
```

```
function F=FaerKiteB(beta,rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)
% Calculates aerodynamic force of kite in spherical coordinate basis
% beta    control angle
% rho,theta,phi     spherical coordinates of kite, rho normalized
% rho1,theta1,phi1  derivatives of spherical coordinates of kite
%                   with respect to dimensionless time
% V0      average windspeed at reference height [m/s]
% l0      characteristic length of tether [m]
% t       time [s]
% airD    density of air [kg/m^3]

Akite=Value('Akite'); %effective wing area of kite [m^2]
AoA0=Value('AoA0'); %angle of attack of the kite in overhead position [deg]

kiteVelocity=[rho1            %dimensionless kite velocity
              rho*theta1
              rho*phi1*sin(theta)];

We=WindSpherical(V0,l0,rho,theta,phi,t)-kiteVelocity; %effective wind
e_w=We/norm(We);  %longitudinal axis of kite

Wep=[0; We(2); We(3)];
e_wp=Wep/norm(Wep);

e_r=[1; 0; 0];
e_sp=cross(e_r,e_wp);
t3=cross(e_w,e_sp);
t3=t3/norm(t3);

AoA= AoA0 + asin(We(1)/norm(We))*180/pi;  %angle of attack of kite [deg]

lift=0.5*airD*Akite*Cl(AoA)*V0^2*norm(We)^2;  %magnitude of lift component
drag=0.5*airD*Akite*Cd(AoA)*V0^2*norm(We)^2;  %magnitude of drag component

%returning the aerodynamic force vector
F=lift*(cos(beta)*t3-sin(beta)*e_sp) + drag*e_w;
end
```

```
function F=FaerTether0(rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)
% Calculates aerodynamic force of straight tethers from effective wind
% at kite and neglecting force along tether, force in spherical coordinate
% basis
% rho,theta,phi     spherical coordinates of kite, rho normalized
% rho1,theta1,phi1  derivatives of spherical coordinates of kite
%                   with respect to dimensionless time
% V0      average windspeed at reference height [m/s]
% l0      characteristic length of tether [m]
% t       time [s]
% airD    density of air [kg/m^3]

tethers=Value('tethers');     %number of tethers
dTether=Value('dTether');     %diameter of one tether [m]
Cnorm=Value('Cnorm');         %normal reaction coefficient

kiteVelocity=[rho1            %relative kite velocity normalized with V0
              rho*theta1
              rho*phi1*sin(theta)];

We=WindSpherical(V0,l0,rho,theta,phi,t)-kiteVelocity;  %effective wind
Wep=[0; We(2); We(3)];
l=l0*rho;    %length of straight tether = radial coordinate of kite

F=1/8*airD*dTether*Cnorm*l*V0^2*norm(Wep)*Wep; %for one tether
F=tethers*F;
end
```

```
function F=FaerTether1(rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)
% Calculates aerodynamic force of straight tethers by integrating
% the forces to tether elements, force in spherical coordinate basis
% rho,theta,phi     spherical coordinates of kite, rho normalized
% rho1,theta1,phi1  derivatives of spherical coordinates of kite
%                   with respect to dimensionless time
% V0      average windspeed at reference height [m/s]
% l0      characteristic length of tether [m]
% t       time [s]
% airD    density of air [kg/m^3]

tethers=Value('tethers');     %number of tethers
dTether=Value('dTether');     %diameter of one tether [m]
Cnorm=Value('Cnorm');         %normal reaction coefficient
Clong=Value('Clong');         %longitudinal reaction coefficient
tol=Value('tolInt');          %tolerance for quadrature

kiteVelocity=[rho1             %relative kite velocity normalized with V0
              rho*theta1
              rho*phi1*sin(theta)];
l=l0*rho;    %length of straight tether = radial coordinate of kite

    function f=LongInt(x)        %integrand function for radial force
        f=zeros(size(x));
        for i=1:length(x)
            wind=WindSpherical(V0,l0,x(i)/l0,theta,phi,t);
                %wind at line element
            effWind=wind(1)-kiteVelocity(1);   %radial effective wind
            f(i)=abs(effWind)*effWind;         %value of integrand
        end
    end

%calculating the radial force component
F(1)=0.5*airD*dTether*Clong*V0^2*quadl(@LongInt,0,l,tol);

    function f=PerpInt1(x) %integrand function for force in theta direction
        f=zeros(size(x));
        for i=1:length(x)
            Wet=WindSpherical(V0,l0,x(i)/l0,theta,phi,t)...
                -[kiteVelocity(1)
                  x(i)/l*kiteVelocity(2)
                  x(i)/l*kiteVelocity(3)];
              %effective wind for tether element
            Wetp=Wet(2:3);
            f(i)=x(i)*norm(Wetp)*Wetp(1);     %value of integrand
        end
    end
```

```
%calculating the force component in theta direction
F(2)=0.5/l*airD*dTether*Cnorm*V0^2*quadl(@PerpInt1,0,l,tol);

    function f=PerpInt2(x) %integrand function for force in phi direction
        f=zeros(size(x));
        for i=1:length(x)
            Wet=WindSpherical(V0,l0,x(i)/l0,theta,phi,t)...
                -[kiteVelocity(1)
                  x(i)/l*kiteVelocity(2)
                  x(i)/l*kiteVelocity(3)];
              %effective wind for tether element
            Wetp=Wet(2:3);
            f(i)=x(i)*norm(Wetp)*Wetp(2);     %value of integrand
        end
    end

%calculating the force component in phi direction
F(3)=0.5/l*airD*dTether*Cnorm*V0^2*quadl(@PerpInt2,0,l,tol);

F=tethers*F';
end


function F=Fgra(rho,theta,l0,m)
% Calculates combined gravity force acting on kite with tethers
% rho,theta,phi     spherical coordinates of kite, rho normalized
% l0      characteristic length of tether [m]
% m       mass of kite [kg]

g=Value('g');               %acceleration due to gravity [m/s^2]
tethers=Value('tethers');   %number of tethers
dTether=Value('dTether');   %diameter of one tether [m]
tetherD=Value('tetherD');   %density of tether material [kg/m^3]
l=l0*rho;          %length of straight tether = radial coordinate of kite

mu=tetherD*pi/4*dTether^2;     %mass per unit length
m1=mu*l/2;                     %effective mass of one tether

F=-(m+tethers*m1)*g*[cos(theta); -sin(theta); 0];
end
```

```
function T=Tension(L,rho,theta,phi,rho1,theta1,phi1,V0,l0,t,airD)
% Calculates combined tension of the tethers with elasticity and wind load
% L       dimensionless tether length
% rho,theta,phi     spherical coordinates of kite, rho normalized
% rho1,theta1,phi1  derivatives of spherical coordinates of kite
%                   with respect to non-dimensional time
% V0      average windspeed at reference height [m/s]
% l0      characteristic length of tether [m]
% t       current time (scalar) [s]
% airD    density of air [kg/m^3]

tethers=Value('tethers');     %number of tethers
dTether=Value('dTether');     %diameter of one tether [m]
Cnorm=Value('Cnorm');         %normal reaction coefficient
E=Value('E');                 %tensile modulus of tether material [Pa]

kiteVelocity=[rho1            %dimensionless kite velocity
              rho*theta1
              rho*phi1*sin(theta)];

We=WindSpherical(V0,l0,rho,theta,phi,t)-kiteVelocity;  %effective wind
Wep=[0; We(2); We(3)];

%calculating the tension
a=(l0*rho)^3/14*(airD/8*tethers*dTether*Cnorm*V0^2*norm(Wep)^2)^2;
b=l0*rho/(E*tethers*pi/4*dTether^2);
c=l0*(L-rho);

X=108*a*b^2-8*c^3+12*sqrt(3)*sqrt(a*(27*a*b^2-4*c^3))*b;
T=(1/6)*(X^(2/3)+4*c^2-2*c*X^(1/3))/(b*X^(1/3));
end
```

System and environment properties

```
function w=WindSpherical(V0,l0,rho,theta,phi,t)
% Calculates dimensionless wind vector in local spherical coordinate basis
% V0    average windspeed at reference altitude [m/s]
% l0    characteristic length of tether [m]
% rho,theta,phi     dimensionless spherical coordinates
% t     time [s]

%calculating cartesian component presentation of the position vector
x=[l0*rho*sin(theta)*cos(phi)
   l0*rho*sin(theta)*sin(phi)
   l0*rho*cos(theta)];

%getting the wind vector
wind=WindProfile(V0,x,t);

%conversion matrix from cartesian to the local spherical coordinate basis
sphericalTransform=[sin(theta)*cos(phi) sin(theta)*sin(phi) cos(theta)
                    cos(theta)*cos(phi) cos(theta)*sin(phi) -sin(theta)
                    -sin(phi) cos(phi) 0];
%converting the wind vector to the local spherical coordinate basis
%and dividing by V0 to make it dimensionless
w=1/V0*sphericalTransform*wind;
```

```
function wind=WindProfile(V0,x,t)
% Calculates wind vector in Earth coordinates
% V0    average windspeed at reference altitude [m/s]
% x     position vector [m]
% t     time [s]

V1=0;       %gust strength [m/s]
omega1=0;  %gust frequency [1/s]
windref=V0+V1*sin(omega1*t); %reference wind speed with sinusoidal gusts

%select wind gradient model by calling either windexp or windlog here
wind=windexp(windref,x); %returning the wind vector
end

function wind=windexp(V0,x) %wind gradient power law
z0=10;      %reference altitude [m]
alfa=1/7;  %Hellmann's exponent

%calculating wind vector
wind=[V0*(x(3)/z0)^alfa
      0
      0];
end

function wind=windlog(V0,x) %logarithmic wind gradient law
Z0=27.5;    %reference altitude [m]
Zr=0.5; %7e-4;    %roughness length [m]

%calculating wind vector
wind=[V0*log(x(3)/Zr)/log(Z0/Zr)
      0
      0];
end
```

```
function value=Value(name)
% Returns values of different kite simulator parameters
% name    name of parameter as string

switch name
    % Kite parameters
    case 'Akite' % effective wing area of kite [m^2]
        value=10;
    case 'AoA0'  % angle of attack of the kite in overhead position [deg]
        value=-2;
    case 'm'     % mass of kite [kg]
        value=4;

    % Tether parameters
    case 'l0'       % characteristic length of tether [m]
        value=100;
    case 'tethers'  % number of tethers
        value=2;
    case 'dTether'  % diameter of one tether [m]
        value=0.003;
    case 'Cnorm'    % normal reaction coefficient
        value=1.2;
    case 'Clong'    % longitudinal reaction coefficient
        value=0.02;
    case 'tetherD'  % density of tether material [kg/m^3]
        value=970;
    case 'E'        % elastic modulus of tether material [Pa]
        value=89e9;

    % Environment parameters
    case 'V0'    % average windspeed at reference height [m/s]
        value=7;   % for windexp
        %value=8;   % for windlog
    case 'g'     % acceleration due to gravity [m/s^2]
        value=9.81;
    case 'airD'  % density of air [kg/m^3]
        value=1.23;

    % Control parameters
    case 'N'         % order of control function definition
        value=1;
    case 'theta0'    % initial theta coordinate in radians
        value=1.1517;
        %value=55*pi/180;
    case 'phi0'      % initial phi coordinate in radians
        %value=5*pi/180;
        value=0;
    case 'thetaMAX'  % safety limit for theta in radians
```

```
        value=80*pi/180;
    case 'timelimit' % limit for dimensionless flight time
        value=10;

    % Solver parameters
    case 'tolInt';   % tolerance for quadrature in tether drag calculation
        value=1e-8;
    case 'tolRel'    % relative tolerance for ODE solver
        value=1e-8;
    case 'tolAbs'    % absolute tolerance for ODE solver
        value=1e-8;
    case 'step'      % initial step for ODE solver
        value=1e-4;
    case 'STEP'      % timespan for each solver run
        value=0.1;

end


function C_L=Cl(AoA)
% Returns lift coefficient of kite as a function of angle of attack [deg]

%C_L=1;     %constant value can be used here instead
           %to simulate constant angle of attack

C_L=0.3+5*pi/180*AoA;
end


function C_D=Cd(AoA)
% Returns drag coefficient of kite as a function of angle of attack [deg]
C_D=0.02+0.08*Cl(AoA)^2;
end
```